Universidade Técnica de Lisboa
Instituto Superior de Economia e Gestão

O Contributo da Análise Orientada para Objectos na Automatização de Sistemas de Informação de Gestão

Mário Fernando Maciel Caldeira

Dissertação para obtenção do grau de Mestre em Gestão sob orientação da Profª. Engª. Ana Maria Ribeiro dos Santos Lucas

Lisboa
1993

Mário Fernando Maciel Caldeira

O Contributo da Análise Orientada para Objectos na Automatização de Sistemas de Informação de Gestão

Dissertação de Mestrado em Gestão

sob orientação da

Profª. Engª. Ana Maria Ribeiro dos Santos Lucas

ISEG, Lisboa, 1993

Agradecimentos

À Profª. Engª. Ana Lucas pela sua preciosa orientação.

Ao Prof. Engº. Henrique Marcelino pelo apoio sempre manifestado, pelas suas sugestões e críticas oportunas.

Ao Prof. Doutor Amílcar Gonçalves pelo permanente incentivo à elaboração da dissertação.

Ao Engº. Amândio Velho pela documentação cedida e disponibilidade para o diálogo sobre tecnologia orientada para objectos.

À Maria Odete Antunes e ao Dr. Artur Cunha pela colaboração na revisão do texto final.

A todos os colegas e amigos que de alguma forma contribuíram para a realização deste trabalho.

ÍNDICE GERAL

Anexos

ÍNDICE DE FIGURAS

SIGLAS UTILIZADAS

AOO - Análise Orientada para Objectos

CASE - Computer Aided Software Engineering

DBR - Desenho Baseado em Responsabilidades

EIS - Executive Information System

INESC - Instituto Nacional de Engenharia de Sistemas e Computadores

OBLOG - Object Logic

OOCASE - Object-Oriented Computer Aided Software Engineering

OMT - Object Modeling Technique

SGBD - Sistema de Gestão de Bases de Dados

SOM - Semantic Object Model

SSA - Structured Systems Analysis

SSADM - Structured Systems Analysis and Design Method

4GL - Four Generation Language (linguagem de 4ª geração)

Introdução

Pensamos que a tecnologia orientada para objectos vai representar, nos anos 90, uma assinalável evolução na concepção e desenvolvimento de sistemas informáticos.

Consideramos igualmente importante abordar a temática da automatização dos sistemas de informação numa perspectiva paralelamente tecnológica[1] e de gestão, procurando estudar o progressivo contributo das novas tecnologias para a eficácia e a eficiência dos sistemas de informação organizacionais.

Este trabalho é uma reflexão crítica sobre o contributo desta nova tecnologia, mais especificamente das suas técnicas de análise, para a automatização de um determinado tipo de sistemas de informação cruciais para o sucesso das organizações - os sistemas de informação de gestão.

1. A escolha do tema

As razões que fundamentalmente conduziram à escolha deste tema foram as seguintes:

a) Importância crescente dos sistemas de informação nas organizações.

A sociedade actual está em permanente desenvolvimento, encontrando-se as organizações envolvidas uma complexa teia de relações sócio-económicas, repleta de informação vital para a sua sobrevivência e crescimento. Paralelamente, a evolução das tecnologias da informação é muito acentuada, permitindo a construção de sistemas com maior grau de complexidade e eficiência.

A importância atribuída aos sistemas de informação tem aumentado à medida que cresce a complexidade das estruturas sociais e se desenvolvem as tecnologias da informação.

---

[1] Este termo é aqui utilizado em sentido restrito, referindo-se às "tecnologia da informação", porque rigorosamente e em sentido mais lato, a tecnologia pode ser definida como a "organização e aplicação de conhecimentos para atingir objectivos práticos" (Zorrinho, 1991, p.49). A própria actividade de gestão incorpora um conjunto de técnicas cientificamente formuladas.

Porter e Millar afirmam que "é difícil subestimar o significado estratégico das novas tecnologias da informação. Esta tecnologia está a transformar a natureza dos produtos, processos, companhias, industrias e até mesmo a competição" (Porter e Millar, 1985, p.149).

b) A informatização das organizações é um problema de gestão.

É imprescindível que a automatização dos sistemas de informação seja concebida de acordo com os objectivos organizacionais, e não tendo apenas como finalidade instalar soluções tecnologicamente avançadas, como frequentemente sucede. A afirmação que Le Moigne fez em 1978 continua válida e significativa em muitas situações: "Para os sistemas de informação a informática não é uma solução, mas um problema" (Le Moigne, 1978a, p.39).

No caso do Sistema de Informação de Gestão, objecto do nosso estudo, o envolvimento dos gestores e responsáveis organizacionais nas fases de planeamento e análise de sistemas de informação é extremamente importante para a funcionalidade do mesmo.

Os custos inerentes à construção de um sistema informático são normalmente relativamente elevados, pelo que o investimento em recursos e tecnologias da informação deve ser cuidadosamente estudado. De acordo com Porter e Millar (1985, p.149), "Os executivos estão cada vez mais cientes que as tecnologias da informação não podem continuar a ser território exclusivo dos Departamentos de Processamento de Dados ou de Sistemas de Informação". A informatização das organizações, na realidade, também é um problema de gestão.

c) Actualidade do tema.

Durante a segunda metade da década de 80, o termo "object-oriented" foi amplamente difundido entre os informáticos.

Inicialmente, este termo estava directamente relacionado com um conjunto de princípios teóricos incorporados em algumas linguagens de programação pouco divulgadas e com utilização relativamente restrita.

Actualmente, a grande maioria das principais linguagens de programação incluem, nas ultimas versões, uma filosofia de "programação orientada para objectos". É igualmente relevante constatar o progressivo aparecimento de Sistemas de Gestão de Bases de Dados Orientados para Objectos.

Mais recentemente, o conceito "orientado para objectos" estendeu-se às técnicas de análise e desenho de sistemas de informação.

d) A tecnologia orientada para objectos pressupõe benefícios na construção de sistemas informáticos.

Apesar da evolução exponencial das tecnologias de suporte físico dos sistemas de informação automatizados (*hardware),* a produtividade no desenvolvimento de aplicações está ainda longe de ser a desejável.

A principal razão explicativa desta situação reside no facto da evolução das técnicas e métodos de análise, desenho e implementação de sistemas, se realizar a um ritmo significativamente mais lento, não permitindo uma resposta atempada à procura existente; é a designada "crise do *software*". Há uma evidente necessidade de encontrar novas formas de desenvolver aplicações com maior rapidez e qualidade (Cox, 1986; Yourdon, 1989).

O conceito de "orientação para objectos" incorpora um conjunto de princípios que se fundamentam numa diferente interpretação da realidade. Como iremos analisar, o aparecimento e adopção deste novo "paradigma" pressupõe benefícios no desenvolvimento de aplicações (Cox, 1986; Taylor, 1991a; Martin e Odell, 1992).

e) Tendência para uma maior concentração de esforços na fase de análise.

Uma grande parte dos problemas com a automatização de sistemas de informação resulta de um mau planeamento ou de uma análise insuficiente (Fisher, 1988). As perspectivas futuras apontam no sentido da concentração de esforços nestas fases, que são fundamentais, para que funcionalmente o sistema de informação esteja enquadrado com os objectivos e requisitos organizacionais. Por outro lado, o desenvolvimento de linguagens de 4ª geração e de produtos CASE (*Computer Aided Software Engineering*), com "ferramentas" para geração automática de código, vai permitir simplificar e acelerar a execução da fase de programação que é tradicionalmente a fase mais longa do ciclo de vida do projecto informático.

f) Inexistência de um modelo uniforme de Análise Orientada para Objectos.

A adopção da tecnologia orientada para objectos na concepção de sistemas de informação levanta alguma discussão.

A comunidade cientifica não apresenta, no momento actual, uma definição globalmente aceite de Sistema de Gestão de Bases de Dados Orientado para Objectos, nem uma perspectiva comum sobre Análise Orientada para Objectos. Este último facto contribui para a existência de uma diversidade de propostas de métodos de análise.

No caso dos sistemas de informação de gestão, alguns autores, como por exemplo Michael Stonebraker (1991), não são a favor da utilização de um modelo de dados e processos orientado para objectos, sugerem apenas a criação de extensões à tecnologia relacional de forma a suportar o armazenamento e processamento de novos tipos de dados (som, fotografia, gráficos complexos, etc.)

2. A terminologia utilizada

Alguns termos, normalmente utilizados em trabalhos desta natureza, apresentam alguma subjectividade na sua interpretação, sendo usual a elaboração de um capítulo introdutório definindo o sentido desses termos. Não vamos seguir esta fórmula devido ao facto de esta dissertação, na sua globalidade, envolver um conjunto de conceitos provenientes de diferentes "escolas". Conceitos esses que julgamos conveniente explicitar, quando necessário, dentro do devido enquadramento contextual, e portanto ao longo do trabalho.

Gostaríamos contudo de assinalar, nesta introdução, um aspecto de carácter linguístico directamente relacionado com a terminologia utilizada e que pensamos ser importante esclarecer.

Está perfeitamente vulgarizado, mesmo junto da comunidade cientifica, a utilização da designação de "metodologias de análise de sistemas". Ao consultarmos o dicionário da língua portuguesa[2], verificamos que a palavra "metodologia" aparece definida como " a parte da lógica que estuda os métodos das diversas ciências, segundo as leis do raciocínio;...", derivando do grego "méthodos" (método) + "lógos" (tratado), logo, tratado dos métodos. Pela nossa interpretação, "metodologia" é ,portanto, uma palavra singular, sem plural.

---

[2] COSTA, J. Almeida e MELO, A. Sampaio, (1991), *Dicionário da Língua Portuguesa*, 6ª edição, Porto, Porto Editora.

No mesmo dicionário, procuramos o significado da palavra método e encontramos a seguinte designação: "programa que antecipadamente regulará uma sequência de operações a executar, com vista a atingir um certo resultado;...; maneira ordenada de fazer as coisas;...".

Pensamos ser mais adequado referirmo-nos a "métodos de desenvolvimento de sistemas" ou "métodos de análise de sistemas", em detrimento da utilização, neste contexto, da palavra "metodologia".

Um método mais geral poderá envolver outros métodos cujo domínio seja mais restrito, assim como implicar a utilização de várias técnicas. Entendemos por técnica, uma forma específica de proceder, através da utilização de conhecimentos, que visa a obtenção de um determinado resultado. Enquanto que o método define a um nível global as etapas a realizar e as técnicas a utilizar, o conceito de técnica tem, em nosso entender, um sentido mais pragmático e uma forte envolvente aplicacional orientada para a resolução do problema.

Uma abordagem cientifica de qualquer tema relacionado com a utilização de tecnologias da informação apresenta também, inevitavelmente, alguns problemas de carácter linguístico, devido ao uso praticamente obrigatório de termos técnicos de origem anglo-saxónica cuja tradução para português se revela por vezes problemática.

Neste trabalho procuramos, na generalidade, utilizar as designações em português desses termos técnicos com base no «Dicionário de Termos Informáticos»[3], publicado pelo Instituto de Linguística Teórica e Computacional, concebido e aperfeiçoado a partir do «Glossário de Termos Informáticos»[4] elaborado pela CT113 (Comissão Técnica de Normalização da Tecnologia Informática). A CT113 depende do Instituto Português de Qualidade que é a autoridade competente para a normalização terminológica nesta área.

Todavia, existem situações em que preferimos utilizar a designação original anglo-saxónica, devido ao facto de esta estar profundamente difundida.

---

[3] Instituto de Linguística Teórica e Computacional, (1993), *Dicionário de Termos Informáticos*, Lisboa, Edições Cosmos.

[4] CT113 (Comissão Técnica de Normalização da Tecnologia Informática), (1991), *Glossário de Termos Informáticos*, Lisboa, Instituto de Informática.

3. Âmbito do trabalho

A utilização em Portugal de métodos de análise orientada para objectos no processo de informatização das organizações é, neste momento, praticamente inexistente. Este facto inviabiliza a realização de um trabalho de campo, com o objectivo de identificar as principais vantagens e dificuldades observadas com a utilização destes métodos, devido à falta de universo que possibilite a obtenção de conclusões válidas.

Pensamos, contudo, que pelo estudo e reflexão da natureza e abordagem propostas em cada um destes métodos, complementadas com algumas (poucas) experiências relatadas da sua utilização a nível internacional, é possível aferir sobre as principais vantagens e inconvenientes da análise orientada para objectos na automatização de sistemas de informação organizacionais e, mais especificamente, de sistemas de informação de gestão.

O facto desta tecnologia estar numa fase de imaturidade, diríamos mesmo "experimental", torna o seu estudo mais difícil, mas simultaneamente mais aliciante.

Este trabalho é, fundamentalmente, uma reflexão crítica sobre o grau de adaptabilidade das principais técnicas e métodos de análise orientada para objectos na concepção e desenvolvimento de sistemas de informação de gestão. Esta reflexão crítica resulta da convergência de três vectores com diferentes origens. O primeiro consiste na natureza e objectivos de um sistema de informação de gestão. O segundo corresponde à definição dos atributos que um método de análise de sistemas de informação deve possuir. O terceiro reside nas características dos principais métodos e pseudo-métodos de análise orientada para objectos.

A estrutura da dissertação engloba cinco capítulos. O primeiro capítulo aborda o conceito de sistema de informação de gestão numa perspectiva organizacional, desenvolvida a partir dos fundamentos da teoria sistémica. O segundo capítulo evidencia as principais fases inerentes ao processo de informatização das organizações, através de uma sucinta abordagem histórica, procurando definir e identificar a fase de análise de sistemas de informação e apresentar os principais problemas com o desenvolvimento de sistemas informáticos que estão na origem da adopção do paradigma da orientação para objectos. O terceiro capítulo foca os conceitos de base inerentes a esse paradigma que são introduzidos na análise de sistemas de informação. No quarto capítulo, define-se uma estrutura de requisitos que, em nosso entender, um método de análise de sistemas de informação orientado para objectos deve contemplar, para ser aplicável na informatização das organizações. Prossegue-se com uma apreciação dos métodos existentes à luz dessa estrutura, tendo por objectivo concluir sobre a sua capacidade de utilização na

automatização de sistemas de informação de gestão. O trabalho culmina com um último e indispensável capítulo de conclusões.

Capítulo I

O Sistema de Informação de Gestão

Nas organizações, é corrente falar-se em "sistemas de informação". A sua relevância parece indiscutível, embora a sua definição e sentido nem sempre se apresentem perfeitamente evidentes.

O objectivo deste capítulo é apresentar o contexto necessário para o enquadramento, identificação e compreensão do conceito de Sistema de Informação de Gestão, objecto do estudo que pretendemos efectuar.

1.1 O conceito de sistema.

A Teoria Geral dos Sistemas, enunciada e definida por Bertalanffy (1973), nasce da tentativa de encontrar um modelo conceptual que una e fundamente as diferentes áreas da ciência.

Esta teoria, cujo campo de aplicação se estende à generalidade das ciências, quer elas sejam formais, naturais ou sociais, vem contribuir de forma significativa para o estudo das organizações e respectivos sistemas de informação.

Segundo Bertalanffy, um sistema pode ser definido, de uma forma simples e intuitivamente acessível, como "um complexo de elementos em interacção" (Bertalanffy, 1973, p.84).

A generalidade das definições de sistema apresentam o conceito tendo em atenção não apenas a sua composição ("um conjunto de elementos em interacção"), mas consideram igualmente relevante a existência de um objectivo para o qual trabalham coordenadamente os diferentes elementos que o constituem. Gómez-Pallete, por exemplo, expressa a essência deste conceito, dizendo que um sistema corresponde a "um conjunto de elementos, relacionados entre si, actuando num determinado ambiente, tendo por finalidade alcançar objectivos comuns, e com capacidade de auto-controlo" (Rivas, 1984, p.61). Esta definição,

como nos diz o autor, não sendo melhor nem pior do que qualquer outra, de entre dezenas geralmente aceites, adapta-se perfeitamente ao estudo dos sistemas de informação.

A capacidade de "auto-controlo" consiste na possibilidade de verificar se o sistema está ou não a atingir os objectivos pré-definidos, de modo' a se introduzirem, quando necessário, as respectivas correcções.

Apesar de eventualmente diferentes na sua essência, os sistemas possuem um conjunto de características e princípios comuns que, pela relevância com que se expressam nas organizações e respectivos sistemas de informação, julgamos conveniente sumariamente enunciar.

Os diferentes elementos que constituem um sistema, na medida em que actuam de uma forma coordenada, permitem que o resultado da sua acção conjunta seja superior, em efeito, ao somatório das suas acções individuais.

Todos os sistemas estão incorporados em outros sistemas (metasistemas) e podem ser sempre divididos em sistemas menores (subsistemas). O estudo de um sistema complexo, ou de grande dimensão, pode ser realizado analisando individualmente os subsistemas que o constituem e a sua respectiva interligação[1]. Entendemos por sistema complexo um sistema constituído por um elevado número elementos, fortemente relacionados entre si.

Com o decorrer do tempo, os sistemas apresentam uma tendência para o aumento da entropia. Considera-se como entropia o processo pelo qual o sistema tende à desorganização e desintegração quando os seus subsistemas não estão devidamente inter-relacionados.

Uma das funções do sistema de informação nas organizações é permitir a comunicação e diminuir a entropia. Como nos diz Bertalanffy, a entropia "é a medida da desordem e por conseguinte a entropia negativa ou informação é a medida da ordem ou da organização, pois esta última comparada com a distribuição ao acaso é um estado improvável." (Bertalanffy, 1973, p.68).

Sistemas abertos, por contraposição a sistemas fechados, são aqueles que apresentam uma forte interacção com o seu ambiente, através de intercâmbios frequentes. Sofrem as consequências das mutações ambientais e provocam, em função da sua própria acção, alterações no meio envolvente. Estes sistemas têm a propriedade de se adaptarem às mutações externas de forma a manterem o seu equilíbrio dinâmico, designado de "homeostase". Quanto mais especializado é o sistema, menor é o seu grau de adaptação a circunstâncias diferentes.

---

[1] Este princípio está subjacente à divisão funcional utilizada pelos métodos de análise estruturada de sistemas de informação.

Segundo Bertalanffy (1973), uma diferença fundamental entre os sistemas fechados e abertos é que os sistemas abertos, também designados de vivos ou orgânicos, podem alcançar um estado estável partindo de diferentes condições iniciais e passando por diferentes caminhos, enquanto que nos sistemas fechados o estado de equilíbrio dinâmico é determinado por condições inicialmente definidas. As restantes características apontadas como pertencentes aos sistemas vivos são, em última instância, consequências deste facto básico. No estado de estável a composição do sistema permanece constante, apesar da contínua troca de componentes com o ambiente.

Na medida em que não existem sistemas completamente isolados do ambiente é usual designarem-se de sistemas fechados aqueles cujas entradas ou saídas são limitadas e perfeitamente previsíveis, funcionando através de uma relação de causa e efeito: são os sistemas mecânicos ou deterministas.

Qualquer sistema em funcionamento é constituído por *inputs* (entradas), processamento e *outputs* (saídas).

Os *inputs* correspondem a tudo aquilo que o sistema recebe para poder operar. O processamento é a parte do sistema que, mediante um determinado objectivo, transforma os *inputs* produzindo os *outputs*. Por último, os *outputs* correspondem ao resultado do processo de transformação ou produto final do processamento.

O nível de recursos necessários à manutenção de um sistema é directamente proporcional à sua dimensão. Este aspecto, como aponta Edward Yourdon (1989, p.34), é muito relevante na automatização de sistemas de informação. Nem sempre, na avaliação dos custos inerentes à utilização de um sistema informático, o custo de manutenção é devidamente ponderado e, segundo Yourdon (1989, p.109), " muitas organizações utilizam 50 a 70% dos seus recursos na manutenção de sistemas antigos". Entende-se por manutenção a actividade que consiste em assegurar a eficiência do sistema, adaptando-o a novos requisitos e corrigindo eventuais anomalias.

1.2 A perspectiva sistémica das organizações sociais.

1.2.1 As organizações sociais

Numa perspectiva sistémica, uma organização pode ser entendida como um sistema aberto em permanente evolução e adaptação às alterações do meio envolvente. Parsons expressa o conceito de organização dizendo que é uma unidade social intencionalmente constituída para atingir objectivos específicos (Parsons, 1960, p.17).

Uma organização é constituída por um conjunto de elementos humanos, materiais e abstractos, que actuam relacionando-se dinamicamente entre si e com o ambiente, na prossecução da sua missão e objectivos.

Na sua estrutura hierárquica de sistemas, Boulding (1956) considera que as organizações sociais são o tipo de sistema mais complexo que se conhece. Cada organização pode ser classificada como um sistema hipercomplexo, resultado de um vasto conjunto de motivações económicas e sociais distintas, mas simultaneamente convergentes, cuja modelação de comportamento, devido ao seu elevado índice de incerteza, requer uma abordagem essencialmente probabilista.

Paralelamente aos objectivos organizacionais, os indivíduos que constituem a organização são dotados de objectivos pessoais. A harmonia entre estes dois tipos de objectivos possibilita um melhor desempenho e permite a realização individual dos elementos que a constituem. O papel decisivo desempenhado pelo factor humano reflecte-se no carácter psico-sociológico da organização e na imprevisibilidade do seu comportamento, fruto de tensões e sinergias internas e da resposta à evolução das variáveis ambientais.

A empresa, na medida em que é uma unidade social organizada com objectivos pré-definidos, é também uma organização. Rogério Ferreira define a empresa como uma "unidade de meios humanos, materiais e financeiros, que actuando segundo imperativos decorrentes das leis do mercado (economia de mercado) ou do Plano (economia planificada), tem como objectivo , através da produção de bens ou serviços, satisfazer as necessidades, quer da comunidade em que está inserida quer dos que nela participam com capital, direcção e trabalho" (Ferreira, 1985, p.22).

Podemos então dizer que uma empresa é um caso particular de uma organização social onde a perspectiva económica assume especial relevância.

Devido à sua importância e carácter multifacetado, as organizações são objecto de estudo da generalidade das ciências sociais.

1.2.2 A gestão das organizações.

Entendemos por gestão a actividade correspondente ao processo de utilização e articulação dos recursos necessários ou disponíveis, para atingir objectivos específicos.

Consideramos a gestão de empresas como "o ramo da gestão que tem por objecto as actividades económicas e empresariais que visam combinar recursos técnicos, humanos e financeiros, de molde a atingir determinados objectivos num determinado contexto ambiental" (Zorrinho, 1991, p.40).

A actividade de gestão consiste fundamentalmente em quatro funções elementares e distintas: planear, organizar, dirigir e controlar.

Na função de planeamento, procura-se identificar os objectivos da organização e formular os planos de acção necessários para a sua realização. A organização (ou estruturação) diz respeito à definição dos diferentes grupos, estrutura hierárquica e relações de trabalho. Por dirigir, entende-se a acção empreendida no sentido de influenciar os membros da organização para a realização das tarefas que lhes estão atribuídas. A função de controlo consiste em verificar se as actividades organizacionais foram executadas de acordo com o planeado.

Apesar de se basear num conjunto sistematizado de conhecimentos, com técnicas e métodos cientificamente formulados, a gestão das organizações não é uma actividade puramente racional, incorporando normalmente uma boa dose de emotividade e intuição, decorrente do seu carácter social, da complexidade das organizações e da dificuldade em prever de uma forma rigorosa as alterações ambientais.

Segundo Hampton, alguns investigadores, depois de observarem gestores em acção, chegaram à conclusão que "na realidade, a gestão é mais reflectida do que reflexiva, mais emoção do que racionalidade e mais caótica do que sistemática" (Hampton, 1983, p.8).

A gestão apresenta um carácter multi-disciplinar, utilizando e interrelacionando técnicas e métodos distintos mas complementares, permitindo uma visão multifacetada e integrada, coadunável com a essência do seu objecto.

1.3 - O sistema organizacional

O sistema organizacional compreende, numa perspectiva estática, as entidades da organização e do ambiente, assim como as diversas associações duráveis entre essas entidades.

Gómez-Pallete considera como entidade " tudo aquilo que existe na realidade e na mente; tudo quanto, sendo tangível ou imaginário, abstracto ou real, é de interesse para a empresa e, mais concretamente, tudo aquilo cujas características anatómicas devem ser conhecidas pela organização" (Rivas, 1984, p.259).

Numa abordagem dinâmica, o sistema organizacional integra a organização como um sistema que processa e transforma bens e serviços, produzindo, com acréscimo de valor, outros bens e serviços, em intensa interligação a montante e a jusante com o seu ambiente (fig.1.1).

Entende-se por "ambiente" o conjunto de elementos do meio envolvente que, apesar de não pertencerem à organização, são susceptíveis de alterar o seu comportamento, ou de serem influenciados pela própria organização (Chiavenato, 1985, p.64); como por exemplo, os clientes, fornecedores, estado ou bancos.

Dentro do sistema "organização" é possível delimitar um conjunto de subsistemas (financeiro, pessoal, produção, ...) com funções bem definidas, que coordenadamente contribuem para o desempenho da missão da organização. A relativa autonomia e especificidade dos dados e processos, inerentes a estes subsistemas, permite a sua individualização e a realização da respectiva gestão com base em técnicas substancialmente distintas. Esta partição funcional da organização reflecte-se na concepção do sistema de informação, implicando a existência de diversos subsistemas de informação, que são normalmente informatizados através de diferentes tecnologias, e nem sempre devidamente interligados, como seria conveniente e desejável.

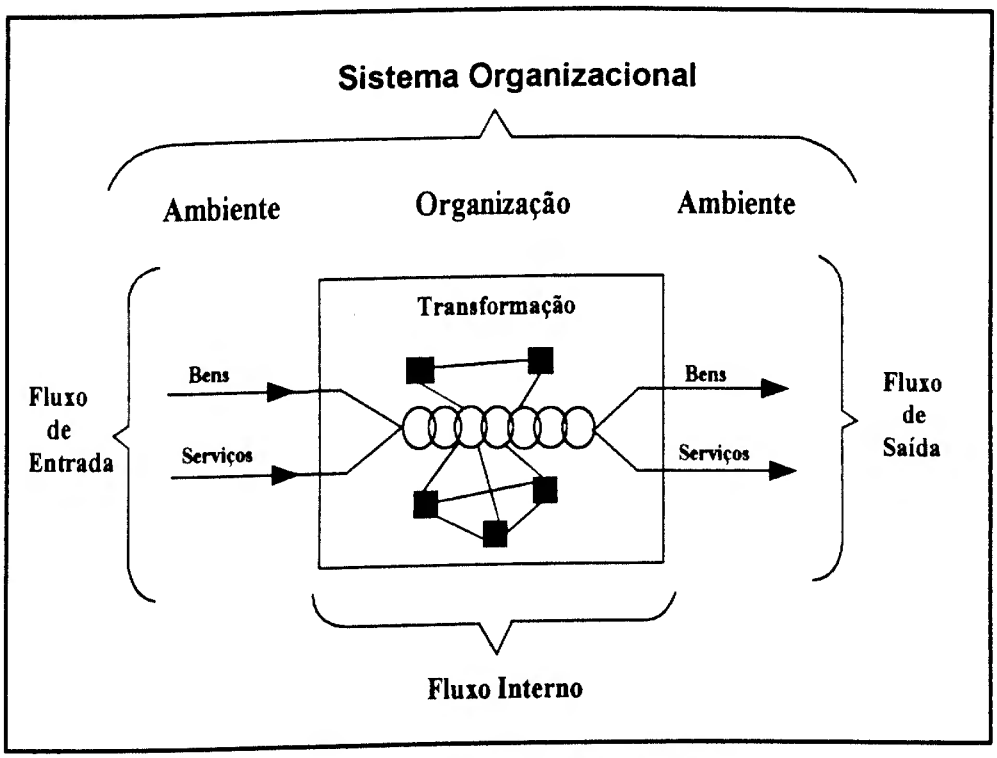


Fig. 1.1 - O Sistema Organizacional

FONTE: Adaptado de Galacsi, *Les systèmes d'information, analyse et conception*, (1986)

Reix (1971a) e Le Moigne (1979a) consideram que qualquer organização é constituída por três subsistemas fundamentais: o sistema de decisão, o sistema de informação e o sistema de operações.

Esta perspectiva da organização é muito importante para a compreensão da função do sistema de informação organizacional.

1.3.1 O sistema de decisão

Herbert Simon (1960) considera que, basicamente, o processo de tomada de decisão se desenvolve interactivamente através de quatro fases distintas: identificação, concepção, selecção e constatação.

A primeira fase, também designada de "inteligência"[2] , corresponde à identificação e compreensão dos factores mais relevantes do problema. Na fase de concepção procede-se à elaboração e avaliação das diversas soluções possíveis. A selecção consiste em escolher a melhor opção e activar a sua execução. Por constatação entende-se a avaliação das decisões anteriormente tomadas.

Sendo assim, podemos resumidamente dizer que a função do sistema de decisão é receber e analisar informação sobre as operações da organização, formulando as diferentes alternativas de operação, seleccionando, promovendo e avaliando, a execução da alternativa considerada mais adequada.

O sistema de decisão corresponde ao conjunto de processos através dos quais a informação é convertida em pré-acção, isto é, na escolha de uma opção que se irá posteriormente procurar executar. Um dos problemas existentes nas organizações advém da dificuldade de transformar rapidamente as decisões em acções.

Segundo Simon (1960) as decisões podem-se subdividir em duas grandes classes: as programadas e as não programadas.

As decisões programadas são decisões de rotina para as quais existe um conjunto de regras fixas e formais, de forma a que a decisão possa ser tomada automaticamente sem necessidade de recorrer ao raciocínio do responsável pela tomada de decisão. Neste caso, é possível representar a decisão de modo formal, sem ambiguidade, através de uma tabela de decisão ou de uma linguagem de programação.

---

[2] Esta designação deve-se ao sentido militar do termo. Simon expressa uma correspondência com os "serviços de inteligência" (serviços de informação), cuja função é captar informação com o objectivo de garantir a segurança.

As decisões não programadas são aquelas em que o processo de resolução não é algorítmico, exige o apelo ao raciocínio, a decisão tem de ser pensada, pois não existe um método formal de resolução. Estas decisões são muito subjectivas e dependem profundamente da intuição do responsável pela decisão, como, por exemplo, a escolha de um argumento publicitário.

Gorry e Scott-Morton (1971) propuseram a classificação das decisões em estruturadas e não estruturadas, como alternativa às designações apresentadas por Simon (decisões programadas e não programadas), pretendendo salientar a essência inerente ao processo de resolução e por entenderem que os termos apresentados por Simon implicavam uma excessiva relação entre a natureza do método de decisão e a utilização de computadores na sua resolução.

Segundo Gorry e Scott-Morton, uma decisão será completamente estruturada quando as respectivas fases de "inteligência", concepção e selecção são estruturadas, e será não estruturada quando nenhuma destas três fases é estruturada.

A divisão entre decisões estruturadas e não estruturadas não é rígida, existindo um conjunto de decisões que poderão ter fases estruturadas e outras não estruturadas, sendo designadas de decisões semi-estruturadas.

Uma decisão semi-estruturada permite, por exemplo, a utilização de um modelo estatístico como auxiliar na resolução do problema, apesar da escolha final, entre as diferentes alternativas possíveis, depender da interpretação do responsável pela tomada de decisão.

A classificação de uma decisão é passível de alterar-se ao longo do tempo, à medida que se aprofunda o conhecimento sobre a natureza do problema e se desenvolvem modelos que o formalizem, paralelamente a métodos científicos para a sua resolução.

Enquanto o processo inerente à tomada de decisões estruturadas é facilmente automatizado, pois é possível desenvolver um modelo formal que o explicite, as decisões não estruturadas são de difícil automatização.

Segundo Gorry e Scott-Morton (1971), os gestores trabalham fundamentalmente com decisões não estruturadas, pelo que a utilização de computadores e sistemas relacionados apresenta algumas limitações na sua aplicação ao processo de tomada de decisão.

Com base em Igor Ansoff, classificamos as decisões de acordo com a sua natureza em estratégicas, tácticas e operacionais (Ansoff, 1977).

As decisões estratégicas estão directamente relacionadas com a definição da missão, objectivos, estratégias e políticas da organização. Estão aqui incluídas as opções fundamentais de aplicação de recursos entre possíveis alternativas, como, por exemplo, a escolha dos produtos a serem fabricados ou a selecção dos mercados onde a empresa irá

operar. São decisões de elevada importância que irão afectar de forma significativa a persecução dos objectivos organizacionais e, por isso, são tomadas pelos gestores de mais alto nível da organização. Uma parte significativa das decisões estratégicas não são estruturadas.

As decisões tácticas dizem respeito à estruturação dos recursos de modo a responder às exigências impostas pela estratégia. Como exemplo, podemos referir as decisões relativas à escolha de canais de distribuição ou à aquisição de equipamento.

As decisões operacionais tem como objectivo maximizar a eficiência do processo de conversão de recursos. Envolvem a fixação de preços, determinação de níveis de *stock*, etc. Incluem um grande número de decisões programadas e o seu nível de importância para a organização é menor que as decisões tácticas ou estratégicas.

A classificação de um determinado tipo de decisão em estratégica, táctica ou operacional, não é linear, envolve um certo grau de subjectividade e depende em muito do impacto dessa decisão nos objectivos organizacionais.

1.3.2 O sistema de operações.

O sistema de operações tem como função a produção de bens materiais e serviços. É o sistema fisico encarregado de realizar as acções correspondentes às decisões elaboradas pelo sistema de decisão e transmitidas através do sistema de informação. Incorpora um conjunto de meios humanos, técnicos e materiais, directamente relacionados com a execução do processo de transformação ou com a realização de serviços relativos à actividade da organização.

A delimitação e identificação do sistema de operações é justificada pelo carácter prático e concretizado das funções organizacionais.

1.3.3 O sistema de informação organizacional.

Concordamos com Gómez-Pallete quando diz que "o termo «informação» continua a pertencer a uma categoria de vocábulos de uso fácil, mas de definição dificil" (Rivas, 1984, p.69).

Numa perspectiva tecnológica, Helder Coelho considera que a informação é o resultado de um processo computacional de manipulação de dados (Coelho, 1986, p.40).

Para Tsichritzis e Lochovsky (1982, p.3), informação é "um incremento de conhecimento que pode ser inferido através dos dados". Esta definição apresenta uma visão complementar da noção anterior, na medida em que pressupõe não só o processo de manipulação dos dados como inclui uma fase de percepção e interpretação, por parte do utilizador, desse conjunto de dados processados.

Henry Lucas Jr. define informação como "uma entidade tangível ou intangível que serve para reduzir a incerteza sobre algum estado ou evento" (Lucas, 1986, p.119). Uma outra definição, apresentada por Le Moigne, considera a informação "um objecto formatado, criado artificialmente pelo homem, tendo por finalidade representar um tipo de acontecimento identificável por ele no mundo real" (Le Moigne, 1978b, p.29).

Le Moigne apresenta o sistema de informação como um sistema que cria, transforma, transmite e memoriza informação, cuja função é fornecer ao sistema de decisão as informações relativas ao sistema organizacional (organização e meio envolvente), necessárias ao funcionamento desse sistema de decisão. O sistema de informação possibilita assim uma ligação dinâmica entre o sistema de operações e o sistema de decisão.

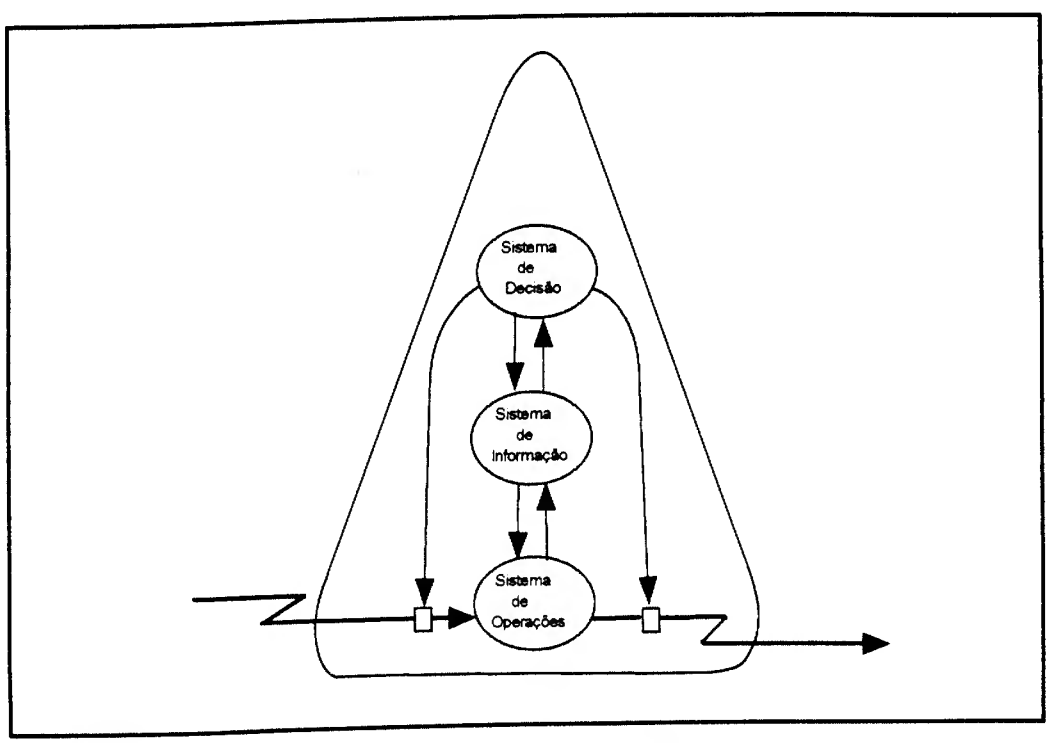


Fig. 1.2 - A função do sistema de informação organizacional.

FONTE: Le Moigne, « La théorie du système d'information organizationnel»,(Le Moigne, 1979a, p.33)

Através da figura 1.2 procuramos, de uma forma gráfica, traduzir simultaneamente a diferenciação e interligação entre os diversos subsistemas da organização, identificando o papel do sistema de informação na captação de informação sobre os "objectos" activos e passivos do sistema organizacional. Consideramos neste contexto por "objecto", todo e qualquer facto ou acontecimento sobre a qual seja relevante guardar informação [3].

Na medida em que as organizações também utilizam a informação como "matéria-prima", ou como produto final, é possível identificar uma parte do sistema de informação que está directamente relacionado com o funcionamento do sistema de operações, independentemente do processo de tomada de decisão. Partindo desta perspectiva, Henrique Marcelino considera o sistema de informação organizacional como " o conjunto de meios e procedimentos, que através de mecanismos de representação, têm como finalidade explicita ou resultado implícito fornecer aos diferentes membros da organização uma percepção do estado e do funcionamento da dita organização e do seu meio envolvente (sistema de informação de gestão) e suportar de modo operacional as actividades do sistema de operações cujo objecto seja informação (sistema de informação produtivo)" (Marcelino, 1990, p.8).

Na realidade, o sistema de informação organizacional, através do seu sub-sistema de informação produtivo, fornece informação sobre a organização e o seu ambiente, não só para os elementos da organização como também para os elementos do meio envolvente (bancos, administração fiscal etc...). O sistema de informação de uma organização é um sistema aberto, com objectivos bem definidos, funcionando em interligação com outros sistemas de informação.

O sistema de informação organizacional deverá possibilitar, aos responsáveis pela decisão, a informação necessária para as decisões programadas, auxiliando na tomada de decisões não programadas, assegurando simultaneamente a comunicação entre os elementos da organização. Para tal, deverá possuir métodos de representação da estrutura e dinâmica do sistema organizacional, assim como um conjunto de meios de tratamento de dados e informação natural.

Por informação natural entendemos as diferentes formas de informação que não tenham sido "artificialmente" estruturadas recorrendo à tecnologia informática. Esta informação pode assumir uma forma oral, textual, gráfica, etc.

---

[3] A noção de objecto em Le Moigne pode se utilizar equiparada ao conceito de entidade, anteriormente apresentado na pág. 12.

Todas as organizações têm obrigatoriamente um sistema de informação. A hipótese da sua não existência põe em causa a comunicação e relacionamento entre os seus elementos e, portanto, o próprio conceito de organização. Gómez-Pallete defende inclusive que "qualquer organização, seja de que tipo for, pode e deve ser interpretada como Sistema de Informação" (Rivas, 1984, p.81), na medida em que pode ser encarada como um sistema que recolhe, memoriza, processa e utiliza informação nos seus processos de decisão e de operação.

Apesar de não existirem organizações sem sistema de informação, nem sempre o sistema existente é eficiente.

O sistema de informação de gestão é o sub-sistema nuclear do sistema de informação organizacional, cujo funcionamento se reflecte significativamente nos resultados da organização, principalmente num ambiente agressivo e mutável. É a parte do sistema de informação, directamente relacionada com o processo de tomada de decisão, que utiliza a informação necessária para a realização da actividade de gestão aos seus níveis estratégico, táctico e operacional.

Robert Anthony (1981)[4] apresentou uma classificação estrutural dos subsistemas de informação da organização, baseada na estratificação da actividade de gestão em planeamento estratégico, controlo de gestão e controlo operacional. Esta estruturação apresenta algum paralelismo com os níveis de decisão identificados por Igor Ansoff.

O planeamento estratégico é entendido como "o processo de decidir sobre os objectivos da organização, as alterações nesses objectivos, os recursos utilizados para alcançar esses objectivos, e as políticas que governam a aquisição, utilização e disposição desses recursos" (Anthony, 1981, p.24). Sendo assim, no âmbito dos sistemas de informação de gestão, os sistemas de informação estratégicos devem possibilitar a informação necessária para a tomada de decisões que afectam significativamente o sucesso da organização, como sejam, decisões de investimento em novos produtos ou mercados, previsões de vendas, previsões de lucro, etc.

O controlo de gestão corresponde ao " processo através do qual os gestores asseguram a obtenção dos recursos e o seu uso efectivo e eficiente na realização dos objectivos organizacionais" (Anthony, 1981, p.27). Os sistemas de controlo de gestão estão directamente relacionados com o controlo orçamental, a análise de vendas, pesquisas de mercado e outros afins.

---

[4] A 1ª edição é datada de 1965.

Por último, o sistema de controlo operacional tem como finalidade assegurar a execução das tarefas de rotina, mais elementares para a organização, como, por exemplo, o processamento de encomendas ou a emissão de facturas.

Entendemos que o sistema de informação de gestão é o subsistema de informação organizacional que suporta a realização da actividade de gestão e que inclui os subsistemas de informação necessários para a realização do planeamento estratégico, controlo de gestão e controlo operacional.

1.4 Sistema informático *versus* sistema de informação.

As tecnologias da informação têm evoluído nos últimos anos de forma muito significativa, permitindo aperfeiçoar o funcionamento do sistema de informação organizacional.

A informatização das organizações possibilita não só um aumento do nível de eficiência do sistema de informação como permite a obtenção de vantagens competitivas em relação aos concorrentes, contribuindo desta forma para o crescimento e desenvolvimento dessa organização.

Designa-se de informatização o processo de automatização recorrendo às tecnologias da informação.

Por tecnologias da informação entendemos o conjunto de processos cognitivos (*software*) e materiais (*hardware*) necessários para a realização de uma actividade de captação, processamento, memorização ou emissão de informação.

A crescente automatização dos sistemas de informação, através da introdução destas novas tecnologias é, de certa forma, responsável pelo facto de existir uma tendência intuitiva, principalmente entre os não especialistas, mas não só, para identificar "sistemas de informação" com "sistemas informáticos"[5].

O sistema informático de uma organização é a parte do sistema de informação cujo tratamento e execução é realizado, parcial ou totalmente, em computador. De uma outra forma, é a componente ou fracção automatizada do sistema de informação. O conceito de sistema de informação é, em nosso entender, um conceito intrinsecamente organizacional,

---

[5] Para tal, também contribui o facto de não existir na língua inglesa uma palavra correspondente ao termo "informática", sendo usual quando se pretende referenciar sistemas informáticos a utilização da expressão "information systems", quando, na verdade, se deveria utilizar a também conhecida expressão "computer based information systems".

que não deve ser confundido com a introdução de tecnologias de base computacional para tratamento de dados, cujo objectivo é permitir a sua automatização mas não constitui a sua essência.

O sistema de informação organizacional não é, portanto, o sistema informático, embora seja fundamental o seu correcto entendimento para que a informatização das organizações se realize de harmonia com os objectivos organizacionais.

O conceito de sistema de informação organizacional, de uma forma geral, e mais especificamente o conceito de sistema de informação de gestão revelam-se de fundamental importância para nós, pois permitem a interligação entre a cultura de gestão e a cultura informática, entre a perspectiva organizacional e a perspectiva tecnológica.

1.5 A automatização do Sistema de Informação de Gestão

Uma assinalável percentagem da utilização de computadores nas organizações está directamente relacionada com tratamento de dados, não envolvendo decisões (Gorry e Scott Morton, 1971). Por outro lado, o conceito tradicional de "Management Information System" (Sistema de Informação de Gestão, numa tradução literal) é normalmente associado apenas aos sistemas transaccionais e operacionais da organização. Subsistindo a ideia, junto de alguns sectores da comunidade informática, que a automatização de sistemas de informação de gestão é plenamente conseguida através da introdução de tecnologias da informação como suporte ao nível do controlo operacional; ou de uma forma mais pragmática, que a automatização do sistema de informação de gestão se completa com a informatização dos processos relativos às actividades operacionais, como registos de encomendas, emissão de facturas, contabilidade, etc.

Os sub-sistemas estratégicos e de controlo de gestão desempenham um papel importante no sistema de informação de gestão, porque captam e fornecem os dados que suportam as decisões críticas para o sucesso organizacional.

Os sistemas de informação estratégicos envolvem um conjunto de decisões essencialmente não estruturadas. Por isso, a introdução de computadores e sistemas informáticos neste tipo de decisões apresenta-se mais difícil, embora não impossível.

O desenvolvimento de "Sistemas de Apoio à Decisão" (*Decision Support Systems*) e de "Sistemas de Informação para Executivos" (*Executive Information Systems*) vem justificar o facto, anteriormente apontado, que os problemas, que inicialmente se apresentam como semi-estruturados ou não estruturados, podem ser estudados e analisados de forma a

se desenvolver e formalizar um modelo que se aproxime da estrutura lógica inerente ao respectivo processo de resolução.

Um Sistema de Apoio à Decisão é um sistema computacional[6] utilizado para auxiliar a tomada de decisão sobre problemas semi-estruturados, cujo processo de resolução é relativamente complexo e requer o tratamento de um grande volume de dados. É função do Sistema de Apoio à Decisão processar esses dados, emitindo resultados sintéticos e objectivos através de *interfaces* agradáveis ao utilizador ("user-friendly") (Reynolds, 1988).

O conceito de *Executive Information System (EIS)* está também associado a um sistema computacional, cuja função é produzir informação pertinente e actualizada para os gestores de topo da organização, em função dos factores críticos de sucesso da respectiva área de decisão. O sistema deve ser personalizado de forma a satisfazer as necessidades individuais desses gestores, cujos requisitos e formas de gestão podem variar significativamente (Reynolds, 1988).

Numa perspectiva funcional, um *EIS* acumula dados provenientes de uma variedade de fontes internas e externas, absorvendo e sintetizando *outputs* de sistemas de controlo de gestão e de controlo operacional.

[6] Entendemos por sistema computacional um sistema que realiza processos de cálculo através da utilização de computadores.

Capítulo II

A concepção e desenvolvimento de sistemas informáticos

A automatização de um sistema (ou subsistema) organizacional, principalmente em organizações de média e grande dimensão, é uma tarefa complexa e difícil, não só de executar como também de gerir.

Uma organização inclui diversos subsistemas que são normalmente automatizados de forma independente através de diferentes projectos, mas mantendo um referencial comum.

O processo de informatização desses subsistemas pode ser descrito através de um ciclo de vida, que normalmente se designa de "ciclo de vida do desenvolvimento de sistemas", "ciclo de vida do desenvolvimento de *software*" ou "ciclo de vida do projecto informático".

Existem vários modelos de representação desse ciclo de vida, expressando diferentes perspectivas de desenvolvimento. A identificação do conjunto de fases que compõem cada modelo está directamente associada ao método de desenvolvimento adoptado.

Apesar de algumas divergências de delimitação e interrelação, a generalidade dos autores reconhecem a existência de um conjunto idêntico de fases principais. Uma destas fases é indiscutivelmente a análise de sistemas de informação[1].

2.1 O ciclo de vida do projecto informático.

Até finais da década de 60, não existia um método formal de desenvolvimento de sistemas de processamento de dados. As aplicações então elaboradas tinham apenas como objectivo automatizar algumas tarefas elementares de nível operacional, como, por exemplo, relatórios de vendas ou emissão de facturas. As funções inerentes ao desenvolvimento de

[1] Esta fase corresponde a uma análise do sistema de informação realizada numa perspectiva informática. Existem, por exemplo, aspectos humanos e comportamentais do sistema de informação que não são tradicionalmente analisados porque não são considerados relevantes para a informatização do sistema organizacional.

sistemas informáticos eram realizadas quase exclusivamente por programadores. As organizações que dispunham de um sistema de informação automatizado estavam no estágio de iniciação[2].

A inexistência de documentação sobre as aplicações em funcionamento era uma constante, o que implicava que futuras alterações estivessem profundamente dependentes dos programadores que construíram essas aplicações. A manutenção e aperfeiçoamento do sistema informático era um problema evidente (Avison e Fitzgerald, 1988).

O programador é um indivíduo fundamentalmente qualificado na codificação de algoritmos através de uma linguagem de programação computacional. É perfeitamente justificável a existência de dificuldades no diálogo com os utilizadores, assim como alguma falta de sensibilidade para compreender, na sua essência, os sistemas a informatizar. Estes motivos, como apontam Avison e Fitzgerald (1988), reflectiam-se no facto de as aplicações realizadas nem sempre satisfazerem as necessidades e expectativas organizacionais.

Com o aumento da complexidade das organizações, paralelamente ao desenvolvimento das tecnologias da informação, cresce a necessidade de conceber e desenvolver sistemas informáticos de maior dimensão e complexidade. Os projectos de informatização envolvem uma vasta equipa de profissionais. A especialização é fundamental, porque permite rapidez de execução e qualidade no produto final. O conhecimento detalhado e documentado dos sistemas de informação a automatizar vem sendo progressivamente valorizado, surgindo então as funções de analista de sistemas e analista-programador.

Em finais da década de 60, o Nacional Computing Centre do Reino Unido recomenda a utilização de um método de desenvolvimento de sistemas com as seguintes etapas: estudo de viabilidade, análise de requisitos, análise de sistemas, desenho de sistemas, programação, revisão e manutenção [3].

Este conjunto de fases, usualmente designado de "ciclo de vida tradicional do desenvolvimento de sistemas", teve um forte impacto junto da comunidade informática (Avison e Fitzgerald, 1988).

---

[2] O estágio de iniciação corresponde ao primeiro dos cinco estágios evolutivos descritos por Nolan em "Managing the crisis in data processing" (Nolan, 1979). Este estágio é caracterizado pelo processamento em *batch* de aplicações operacionais com o objectivo de reduzir custos.

[3] As fases inerentes a este método estão detalhadamente descritas em DANIELS, A. e YEATS, D. A., (1971), *Basic Traning in System Analysis*, Londres, Pitman.

Várias variantes do ciclo de vida, próximas do modelo anterior, são apresentadas por diferentes autores. O diagrama que apresentamos na figura 2.1, introduzido por Royce em 1970 [4] e designado de "modelo em cascata de desenvolvimento de sistemas", ficou célebre.

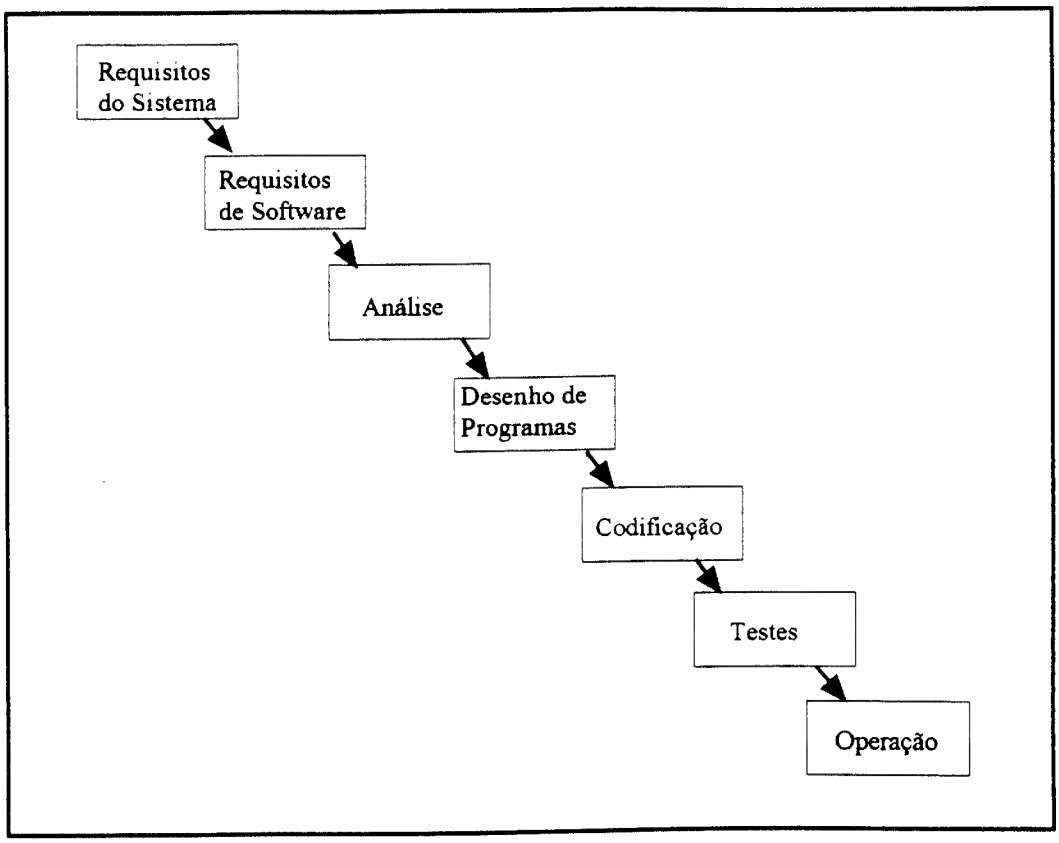


Fig. 2.1 - O modelo em cascata de desenvolvimento de sistemas.

FONTE: Royce, Wiston, « Managing the Development of Large Software Systems », (Royce, 1970, p.2)

A designação de "modelo em cascata" advém do facto de se considerar o desenvolvimento de sistemas como um processo linear, dividido num número de fases consecutivas e sequencialmente realizadas.

Uma representação mais perfeita e dinâmica é exibida na fig. 2.2. O processo de desenvolvimento é apresentado como interactivo e com possibilidade de retroacção, na medida em que na fase de teste o sistema pode ser considerado inadequado, sendo necessário a sua revisão, e durante a manutenção aparecem normalmente deficiências de programação ou surgem novos requisitos, que implicam alterações nas especificações dos programas ou levam à concepção de um novo sistema, reiniciando-se o ciclo.

[4] ROYCE, Winston W., (Agosto 1970), « Managing the Development of Large Software Systems», *IEEE Proceedings*, pp. 1 a 9.

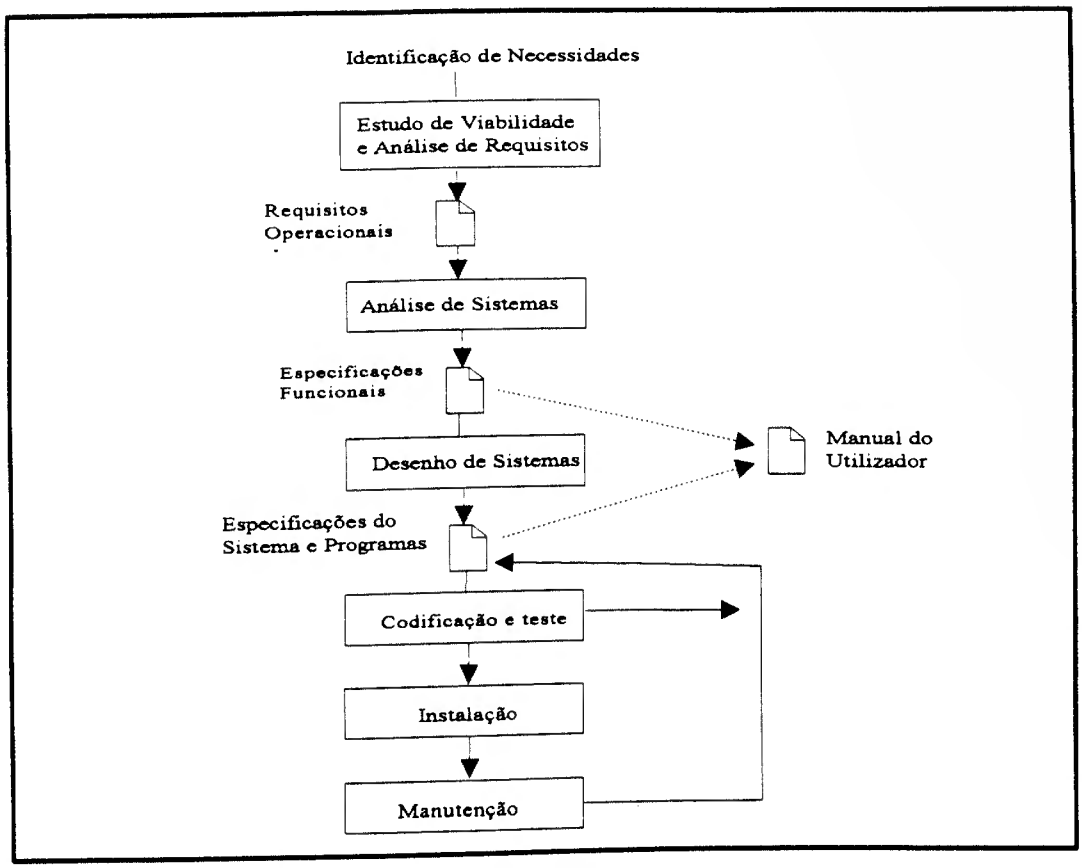


Fig. 2.2 - O ciclo de vida do desenvolvimento de sistemas

FONTE: Adaptado de Guton, *Inside Information Technology* (Guton, 1990, p.223)

As principais fases do ciclo de vida tradicional permanecem comuns à maioria dos métodos actuais de desenvolvimento de sistemas informáticos, apesar da filosofia de desenvolvimento e das técnicas utilizadas em cada uma dessas fases terem evoluído significativamente. As longas especificações funcionais deram lugar às especificações estruturadas. Surgiram técnicas de análise de sistemas como os diagramas de fluxos de dados e o modelo entidade-associação[5]. As linguagens tradicionais de 3ª geração, como o

[5] O diagrama de fluxos de dados é uma técnica que expressa a forma como circula a informação no (sub)sistema de informação em estudo. Esta técnica foi inicialmente apresentada por Tom De Marco (De Marco, 1978). O modelo entidade-associação representa a estrutura de dados de um determinado (sub)sistema e tem a sua origem num artigo publicado por Peter Chen em 1976 (Chen,1976).

COBOL, são progressivamente substituídas por potentes linguagens de programação (4GLs [6], C, etc.).

A utilização de produtos CASE (*Computer Aided Software Engineering*) permite reduzir o tempo de desenvolvimento, assim como flexibilizar a realização de alterações necessárias nas especificações do sistema. Representa uma tendência evolutiva de aplicação das modernas tecnologias da informação à generalidade das fases do ciclo de vida do desenvolvimento de sistemas. O seu eficiente campo de aplicação centra-se fundamentalmente nas fases de análise e desenho, com possíveis extensões ao planeamento estratégico ("upper case") e geração automática de código ("lower case").

A evolução desta tecnologia, no sentido de gerar código executável a partir das mini-especificações do sistema de informação, reflecte-se numa crescente importância atribuída à análise de sistemas, em detrimento da fase de programação (tradicionalmente mais relevante).

As técnicas estruturadas caracterizam-se por uma divisão modular, baseada no princípio geral de que qualquer sistema se pode dividir em sub-sistemas, e num desenvolvimento "top-down", em sentido hierarquicamente descendente, do geral para o particular. Sendo assim, estudam-se, desenham-se, ou constroem-se (depende da fase) inicialmente os módulos principais e só depois os respectivos sub-módulos [7].

As vantagens atribuídas a estas técnicas concretizam-se num aumento de produtividade e diminuição dos custos de manutenção, devido ao facto de os sistemas apresentarem um maior grau de *fiabilidade*[8] e os erros encontrados serem de menor relevância (Yourdon, 1986).

Tendo por base a utilização de técnicas estruturadas de análise, desenho e programação de sistemas de informação, Yourdon (1988) apresenta de forma detalhada o ciclo de vida do projecto informático. Considera este ciclo como um conjunto interactivo de actividades que, apesar de algumas precedências evidenciadas, não são obrigatoriamente realizadas de forma sequencial, sendo possível iniciar uma nova actividade sem a conclusão da actividade supostamente anterior.

---

[6] 4GL ("Four Generation Language") corresponde a uma linguagem de programação que aceita especificações sobre os requisitos do sistema como *input* e gera um programa de acordo com esses requisitos. As linguagens de 4ª geração dispõem normalmente de geradores de: menus, relatórios, ecrans, etc.

[7] Uma perspectiva global dos princípios subjacentes às técnicas estruturadas é apresentada em YOURDON, Edward, (1986), *Managing the Structured Techniques*, Englewood Cliffs, Prentice-Hall.

[8] Entendemos por *fiabilidade* a capacidade de os sistemas não apresentarem erros, de serem fiáveis.

Em nossa opinião, o diagrama de fluxos de dados (fig. 2.3) é utilizado de uma forma feliz para representar o ciclo de vida, pois permite, simultaneamente, observar a interdependência entre as diferentes fases, aferir sobre a participação das entidades externas[9] no desenvolvimento do projecto e identificar quais os documentos que circulam entre essas fases, apresentando uma orientação de fluxos sem referência temporal.

Apesar da excelência da técnica em questão, a representação da fig. 2.3 não contempla os fluxos relativos à retroacção. Na verdade, a realização de praticamente qualquer actividade do ciclo pode implicar revisões em actividades anteriores, pelo que seria correcto assinalar os respectivos fluxos. Tal não acontece porque a densidade gráfica resultante iria afectar a clareza da descrição e, portanto, a sua compreensão. Na figura citada estão apenas assinalados os fluxos de dados mais frequentes e relevantes.

As fases ou actividades apresentadas por Yourdon (1988) para o projecto estruturado e que vamos seguidamente descrever de forma resumida, com o intuito de identificar e delimitar a fase de análise, são as seguintes:

- Estudo de viabilidade,
- Análise,
- Desenho,
- Programação,
- Geração de testes,
- Teste final,
- Descrição de procedimentos,
- Conversão de bases de dados,
- Instalação.

[9] Uma entidade externa corresponde a um elemento fora do âmbito do estudo, mas que realiza com este um intercâmbio de informação.

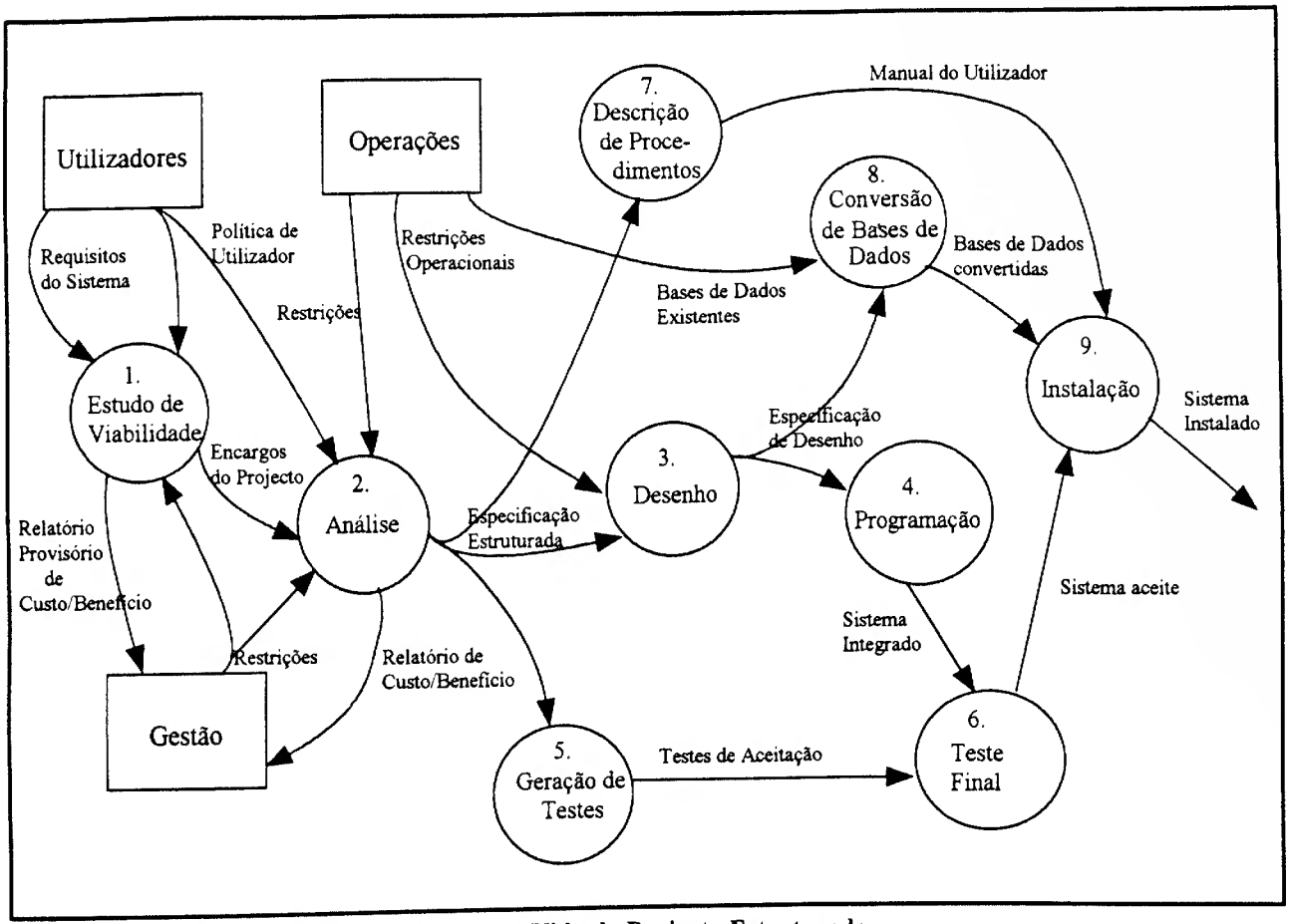
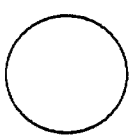


Fig. 2.3 - O Ciclo de Vida do Projecto Estruturado

FONTE: Adaptado de Yourdon, *Administrando o Ciclo de Vida do Sistema*, (Yourdon, 1989a, p.54)

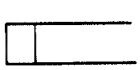
Notação utilizada:

Entidade Externa - Elemento fora do âmbito do estudo, mas que realiza com este um intercâmbio de informação.



Processo - Representa uma actividade de processamento ou tratamento de dados, transformando fluxos de entrada em fluxos de saída.

Fluxo de dados - Representa a circulação de informação, quer esta se apresente sob uma forma documental, oral ou simbólica.



Arquivo - Corresponde a um depósito permanente ou temporário de dados.

O estudo de viabilidade inicia-se quando o sistema de informação actual não corresponde às necessidades organizacionais e os utilizadores sugerem a automatização de uma ou mais partes do seu trabalho.

Entendemos por utilizadores os indivíduos da organização que, devido ao desempenho das suas funções estratégicas, tácticas ou operacionais, têm legitimidade para aferir sobre a eficiência e eficácia do sistema de informação, quer ele seja automatizado ou não.

Nesta primeira fase, procura-se sumariamente identificar as deficiências do sistema actual e estabelecer alternativas, manuais ou computáveis, viáveis para a sua resolução. Para cada alternativa, procede-se à avaliação dos recursos técnicos, humanos e económicos necessários, elaborando um relatório custo/benefício que será enviado para os responsáveis pela gestão da organização.

A decisão final de escolha de uma opção, ou de simplesmente abandonar ou adiar o projecto, caberá logicamente à gestão, com base no relatório custo/benefício (fig 2.3) e nas suas próprias restrições, normalmente de ordem financeira.

A fase de análise é considerada crítica para o sucesso do projecto. Consiste num estudo detalhado do funcionamento do subsistema de informação a automatizar. Utilizam-se várias técnicas, cuja interligação e complementaridade permitem uma visão integrada do sistema de informação. Essas técnicas, que apresentam semelhanças na maioria dos métodos de análise estruturada, procuram incorporar uma perspectiva funcional, estrutural e evolutiva do sistema em análise.

Na fase de desenho procede-se à identificação dos módulos de *software* e *interfaces*, correspondentes às especificações obtidas na fase de análise. A distribuição dos módulos pelos diferentes processadores é realizada recorrendo a diagramas de estrutura.

É igualmente nesta fase que se selecciona qual o suporte físico (*hardware*) mais adequado e o *software* de base a utilizar (sistema operativo, linguagens de programação, Sistemas de Gestão de Bases de Dados, etc.).

A programação consiste na codificação, através de uma linguagem de programação, das especificações anteriormente produzidas. A partir dessas especificações, é importante conceber um conjunto de testes que permitam verificar a eficiência do sistema informático.

O teste de aceitação corresponde à execução dos testes anteriormente concebidos, para aferir sobre a qualidade desse sistema e decidir se este está, ou não, em condições de ser instalado.

Se houver uma alteração substancial no sistema informático existente poderá ser necessário a realização de um processo de conversão de dados. Esta actividade será

eventualmente demorada, caso se tenha de converter um grande volume de dados para um formato inteiramente diferente do anterior, ou poderá simplesmente não existir.

A fase de instalação ocorre quando o sistema informático é declarado oficialmente como operacional e é colocado em funcionamento, terminando então o projecto.

Após a instalação do sistema informático, passa-se para uma outra fase complementar ao desenvolvimento - a manutenção. A manutenção do sistema, na tentativa de o reajustar a novos requisitos ou corrigindo eventuais deficiências, poderá dar início a um novo ciclo de desenvolvimento.

A definição das características globais do sistema informático de uma organização pode e deve ter origem num trabalho de planeamento estratégico de sistemas de informação. Em organizações de pequena dimensão ou superficialmente informatizadas, não é usual o planeamento estratégico do sistema de informação, iniciando-se a informatização (ou "reinformatização") do sistema existente através de uma verificação intuitiva de que este não satisfaz as necessidades emergentes.

O planeamento estratégico de sistemas de informação é uma tarefa de gestão, directamente relacionada com a integração dos sistemas de informação no processo de planeamento organizacional, e que procura analisar como é que a informação e as tecnologias da informação podem contribuir para alcançar os objectivos estratégicos.

2.2 Definição e delimitação da fase de análise de sistemas de informação.

A análise de sistemas corresponde a um estudo detalhado do funcionamento de um sistema (ou subsistema) de informação, com o intuito de o aperfeiçoar funcionalmente, normalmente através da sua informatização.

Enquanto a análise de sistemas de informação se centra na eficiência do sistema de informação, o planeamento estratégico de sistemas de informação preocupa-se com a eficácia do sistema "organização". Esta diferença resulta de um posicionamento conceptual distinto. A perspectiva global e corporativa, inerente ao planeamento de sistemas de informação, implica a identificação de entidades mais abrangentes, que poderíamos designar de "macro-entidades" quando comparadas com as provenientes da modelação detalhada de subsistemas bem delimitados, realizada durante a fase de análise.

Não existe uma distinção globalmente aceite e indiscutível entre análise e desenho de sistemas de informação. Alguns modelos de concepção e desenvolvimento de sistemas de informação estão extremamente orientados no sentido da construção de *software* e não

reconhecem a existência de uma fase de análise, caminhando rapidamente de uma breve e sumária especificação de requisitos para o desenho de um sistema computacional.

Entendemos, em concordância com Henderson-Sellers (1992, p.150), que a razão da separação entre análise e desenho está relacionada com o objectivo da modelação em cada um dos níveis. A análise de sistemas procura representar, através de um modelo, uma parte do mundo real, na perspectiva de funcionamento do sistema de informação, independentemente dos detalhes de implementação. O desenho corresponde a um processo de criação de um modelo computacional baseado no modelo do real anteriormente construído.

Pensamos que um método de análise de sistemas de informação deverá ter como objectivo a representação de um dado sistema de informação, através da construção de um modelo que simplifique a complexidade inerente à natureza desse sistema, salientando os seus aspectos relevantes do ponto de vista estrutural, funcional e evolutivo, e identificando a informação necessária ao seu funcionamento.

O método deverá apresentar um conjunto de etapas interligadas que explicitem os diferentes passos a seguir no processo de estudo do sistema de informação e incluir técnicas que possibilitem a realização desse estudo.

A notação, gráfica ou textual, a utilizar pelas diferentes técnicas, deve possibilitar a gestão da complexidade, permitindo a visualização global e parcial desse sistema de informação através de vários níveis de detalhe.

É igualmente fundamental a existência de uma estrutura global de conceitos e de regras de construção associadas às técnicas de análise.

Na medida em que as tecnologias da informação contribuem de forma significativa para aumentar a eficácia e eficiência do sistema de informação organizacional, é importante (mas não condição *sine qua non*) que o método de análise de sistemas de informação esteja integrado, ou seja possível a sua integração, num método de maior amplitude que cubra as restantes fases do ciclo de vida do projecto de concepção e desenvolvimento de sistemas de informação automatizados.

2.3 Métodos e técnicas de análise estruturada de sistemas de informação.

De entre as dezenas de métodos de análise e concepção estruturada de sistemas de informação actualmente existentes, vamos abordar alguns que julgamos mais relevantes para caracterizar os principais aspectos relativos à fase de análise.

Nem todos os métodos aqui apresentados cumprem, de uma forma rigorosa e formal, os requisitos evidenciados no ponto anterior. Contudo, os seus propósitos são orientados para o mesmo objectivo: a análise e concepção de sistemas de informação.

O método SSA ("Structured Systems Analysis"), baseado principalmente no trabalho de Gane e Sarson (1979), apresenta uma abordagem fundamentalmente funcional, mas também estrutural, do sistema de informação.

A principal técnica utilizada, e intensivamente descrita por De Marco (1978), é o diagrama de fluxos de dados (DFD). O DFD permite representar um modelo de circulação da informação no sistema em estudo e tem como elementos integrantes: processos, arquivos de dados, entidades externas e fluxos de dados.

Para além da técnica anteriormente citada (DFD), o método contempla a construção e utilização de um dicionário de dados, que consiste numa descrição detalhada dos diferentes elementos do sistema, e inclui a definição e normalização das estruturas de dados inerentes aos arquivos.

A primeira etapa, sugerida pelo método, consiste num estudo inicial do problema, com identificação das funcionalidades que o novo sistema de informação automatizado deverá cobrir. Após o estudo inicial, seguir-se-à um estudo detalhado, do ponto de vista físico e lógico, do sistema de informação existente e a definição de diferentes alternativas de solução que levarão ao desenho físico do novo sistema informático.

Da evolução do SSA resultou um método mais recente de análise estruturada, proposto em McMenamin e Palmer (1984) e Yourdon (1989), designado de "Análise Essencial de Sistemas" (*Essential Systems Analysis*). Este método surge como resultado da constatação de que a construção dos modelos físicos e lógicos do sistema existente, anteriormente recomendada, consome tempo excessivo, o que poderá implicar que, quando o novo sistema estiver terminado, este possa estar desactualizado. Por outro lado, a necessidade crescente de sistemas informáticos em tempo-real contribuiu para a concepção e utilização de técnicas que possibilitam uma representação da evolução em função do tempo, como os diagramas de transição de estados, e a inclusão nos DFDs de fluxos e processos de controle.

A análise estruturada clássica, proposta no SSA, baseia-se fundamentalmente numa descrição funcional do sistema de informação, realizando-se a modelação de estruturas de dados[10] de uma forma rudimentar, devido à importância secundária que lhe é atribuída. Com a evolução dos sistemas organizacionais e a crescente utilização de Sistemas de Gestão de Bases de Dados relacionais, surge a necessidade de incluir uma técnica que possibilite a modelação de sistemas de informação com relativamente complexas relações de dados.

A Análise Essencial de Sistemas refere a criação de um "modelo essencial", onde se define o que o futuro sistema de informação deve contemplar, independentemente dos aspectos tecnológicos referentes à sua automatização. O "modelo essencial" é composto de dois modelos principais: o "modelo ambiental" que representa a interligação e dependência do sistema de informação em relação ao meio envolvente e o "modelo comportamental" que expressa o seu comportamento interno de forma a interagir com sucesso com o ambiente.

O modelo comportamental apresenta uma visão tridimensional do sistema de informação incorporando complementarmente os diagramas de fluxos de dados, o modelo entidade-associação e os diagramas de transição de estados.

O diagrama de fluxos de dados baseia-se numa perspectiva funcional e possibilita a decomposição do sistema de informação em diferentes níveis de detalhe, de forma a facilitar o seu estudo. As especificações dos processos descrevem logicamente os detalhes de cada função. O modelo entidade-associação expressa a estrutura de dados existente e o diagrama de transição de estados transmite uma visão dinâmica das entidades. Salienta-se também a importância da elaboração de um dicionário de dados, que consiste numa referência descritiva dos diferentes elementos envolvidos na fase de análise.

O método SSADM ("Structured Systems Analysis and Design Method") foi desenvolvido, a partir de 1980, no Reino Unido, pelos consultores Learmonth and Burchett Management Systems em colaboração com a agência oficial CCTA ("Central Computer and Telecommunications Agency") (Downs *et al*, 1992).

O SSADM apresenta um conjunto detalhado de passos e tarefas bem definidas, a executar para as fases de estudo de viabilidade, análise e desenho de sistemas.

A análise de sistemas de informação fundamenta-se, a exemplo dos métodos anteriores, em três modelos, designados de modelo de fluxos de dados, modelo lógico de dados e modelo entidade-eventos.

---

[10] É frequente o uso da expressão "modelização de dados". Acontece, porém, que o presumível verbo "modelizar" não existe na lingua portuguesa, pelo que utilizamos o verbo "modelar" no sentido de: representar através de um modelo.

O modelo de fluxos de dados baseia-se na elaboração de DFDs, que exibem na versão 4 do SSADM uma notação extremamente desenvolvida, incluindo, por exemplo, fluxos físicos de bens e a distinção entre diferentes tipos de arquivos de dados.

O modelo lógico de dados apresenta grande capacidade de expressão e é complementado com a análise relacional de dados que sugere, tal como nos métodos anteriores, a normalização de tabelas na terceira Forma Normal [11].

Por fim, o modelo entidade-eventos que apresenta capacidade de interligação com os outros dois modelos permite observar quais os eventos que podem influenciar o sistema de informação. Um evento é entendido como algo que ocorre num determinado momento no tempo e que é susceptível de afectar o estado das entidades. O modelo entidade-eventos apresenta duas técnicas diferenciadas pela perspectiva com que representam a mesma realidade: o ciclo de vida das entidades[12] e o diagrama de efeitos correspondentes.

O ciclo de vida das entidades expressa a evolução das entidades em função da ocorrência de eventos. O diagrama de efeitos correspondentes expõe, para cada evento, quais as entidades que por ele são afectadas.

O método *Information Engineering*, da autoria de James Martin e Clive Finkelstein, apresenta quatro grandes etapas: planeamento estratégico da informação, análise, desenho e construção (Martin, 1986).

Consideramos relevante, neste método de concepção e desenvolvimento, a inclusão de uma fase de planeamento estratégico da informação (onde se definem objectivos, oportunidades e ameaças, factores críticos de sucesso e estratégias para os sistemas de informação) e a respectiva interligação com a fase de análise. Utilizam-se, ao longo do método, várias técnicas de análise, idênticas ou com funções semelhantes às anteriormente descritas, como os diagramas de fluxos de dados e o modelo entidade-associação.

Outros métodos, como o MERISE (Tardieu *et al.*, 1984), o AXIAL (Pellaumail, 1986) ou o SADT ("Structured Analysis and Design Technique") (Ross, 1977), também se baseiam numa abordagem estática, do tipo "entidade-associação", paralelamente a uma abordagem dinâmica através da representação de operações e eventos.

O MERISE, de origem francesa, abrange o ciclo de vida desde a fase de concepção, que se inicia com a definição de um plano director, até aos aspectos inerentes à manutenção.

---

[11] Aplicação das "Formas Normais" definidas em CODD, E.F., (Junho 1970), « A Relational Model of Data for Large Shared Data Banks », *Communications of the ACM*, Volume 13, Nº6, e DATE, C.J., (1990), *An Introdution to Database Systems*, Volume 1, 5ª edição, Reading, Addison-Wesley.

[12] A designação original desta técnica no SSADM é "Entity Life Histories". A designação de "ciclo de vida das entidades" foi adoptada em Portugal pelo Instituto Nacional de Administração.

A análise de sistemas é documentada através de um "modelo conceptual de tratamentos", concebido a partir da identificação dos fluxos existentes entre os principais elementos do sistema de informação em estudo. A perspectiva funcional é concluída com a elaboração do "modelo organizacional de tratamentos" que complementa o modelo anterior com a identificação dos processadores do sistema de informação, transmitindo simultaneamente uma referência temporal a esse modelo. A representação estática e estrutural é realizada através da construção do "modelo conceptual de dados" e do "modelo lógico de dados".

Globalmente, podemos dizer que os métodos de análise e concepção estruturada de sistemas de informação apresentam, na sua generalidade, como suporte fundamental, uma técnica de representação funcional da circulação de informação no sistema em estudo - o diagrama de fluxos de dados. Uma função importante atribuída a esta técnica é a possibilidade de assimilar o princípio básico da teoria geral dos sistemas em que um sistema pode ser sempre subdividido em sistemas menores (subsistemas) e incorporado em sistemas de maior dimensão (metasistemas). O DFD permite uma decomposição funcional do sistema de informação em diferentes níveis, o que se revela extremamente importante para a compreensão desse sistema e para a gestão do projecto de informatização. O DFD é, em nossa opinião, uma excelente técnica de gestão da complexidade.

A visão dinâmica do processamento de dados, proporcionada pelos DFDs, deve ser complementada com a perspectiva estrutural. Neste sentido, os métodos de análise adoptaram um conjunto de técnicas inicialmente definidas para a modelação de bases de dados, como o modelo entidade-associação e a normalização de tabelas.

A necessidade de assinalar a permanente alteração, repercutida sobre o sistema de informação, em função da evolução temporal e da ocorrência de acontecimentos internos ou externos a esse sistema, levou à inclusão de técnicas como o ciclo de vida das entidades (no SSADM) ou os diagramas de transição de estados (na "Análise Essencial de Sistemas").

2.4 Desenvolvimento acelerado através da rápida construção de protótipos.

O desenvolvimento de sistemas informáticos em áreas que envolvam novos métodos de operação ou gestão, em que os utilizadores são incapazes de transmitir de forma fidedigna os requisitos necessários ao funcionamento do sistema de informação, assim como a necessidade de produzir *software* num curto espaço de tempo, justificam a utilização de protótipos.

Um protótipo é um sistema que é rapidamente desenvolvido, através de uma versão que podemos designar de experimental, contemplando apenas os aspectos mais relevantes, e que será progressivamente refinado e aperfeiçoado.

O protótipo poderá funcionar como uma versão de teste, a partir da qual se podem extrair conhecimentos úteis para a elaboração do sistema final que posteriormente irá ser construído, ou concebido com o intuito de, ao ser melhorado, resultar na versão final.

A construção de *software* através da rápida criação de protótipos (*rapid prototyping*) representa uma diferente filosofia de desenvolvimento, pois procura-se desenhar e construir o mais rapidamente possível, atendendo apenas às principais especificações, não se efectuando inicialmente um estudo detalhado do sistema de informação.

Uma vantagem na utilização de protótipos é que o protótipo serve de meio de comunicação entre a equipa de desenvolvimento e os utilizadores, permitindo mais facilmente expressar os detalhes relativos ao sistema informático em desenvolvimento do que as especificações gráficas ou textuais (Jacobson *et. al.*, 1992, p.27). A utilização de protótipos requer uma participação activa dos utilizadores.

Um modelo designado de "desenvolvimento em espiral" foi apresentado por Boehm (1988). Este modelo considera o desenvolvimento de um sistema informático como um processo continuo, representado graficamente através de uma espiral que expressa a realização de várias versões consecutivas desse sistema informático, no sentido de efectuar o seu constante aperfeiçoamento.

O modelo em espiral, concebido com uma orientação para a avaliação do risco, está vocacionado para o desenvolvimento através da rápida construção de protótipos (Boehm, 1988, p.65).

Cada ciclo do modelo em espiral começa pela identificação dos objectivos de *performance*, funcionalidade e capacidade de alteração, para a versão a obter no final desse ciclo. É igualmente importante a definição das alternativas de implementação e restrições impostas em termos de custo, tempo e *interfaces* com o utilizador. Nos passos seguintes, procura-se avaliar as diferentes alternativas de implementação em relação às restrições, o que eventualmente levará à identificação de áreas de risco para o projecto. O nível e tipo de risco irá condicionar o protótipo a desenvolver. Se o protótipo obtido for operacionalmente satisfatório e possibilitar a evolução futura desse sistema computacional, procede-se ao seu aperfeiçoamento gradual, através da construção de novos protótipos, até se obter o sistema final desejado.

Segundo Boehm (1988), o modelo em cascata, anteriormente apresentado, não é adequado para referenciar a evolução possibilitada pela rápida construção de protótipos e utilização de linguagens de 4ª geração.

De acordo com Boehm (1988, p.66), o modelo em espiral apresenta uma série de vantagens adicionais em relação aos modelos anteriores, na medida em que :

- Contempla a possibilidade de reutilização do *software* existente;
- Facilita o processo de evolução e alteração do *software*;
- Introduz um mecanismo de definição de objectivos de qualidade para o desenvolvimento de *software*;
- Baseia-se na eliminação prévia de erros e alternativas inadequadas;
- Aborda de uma forma conjunta o desenvolvimento e manutenção do *software*;
- Possibilita um método de integração do desenvolvimento de *software* e *hardware*.

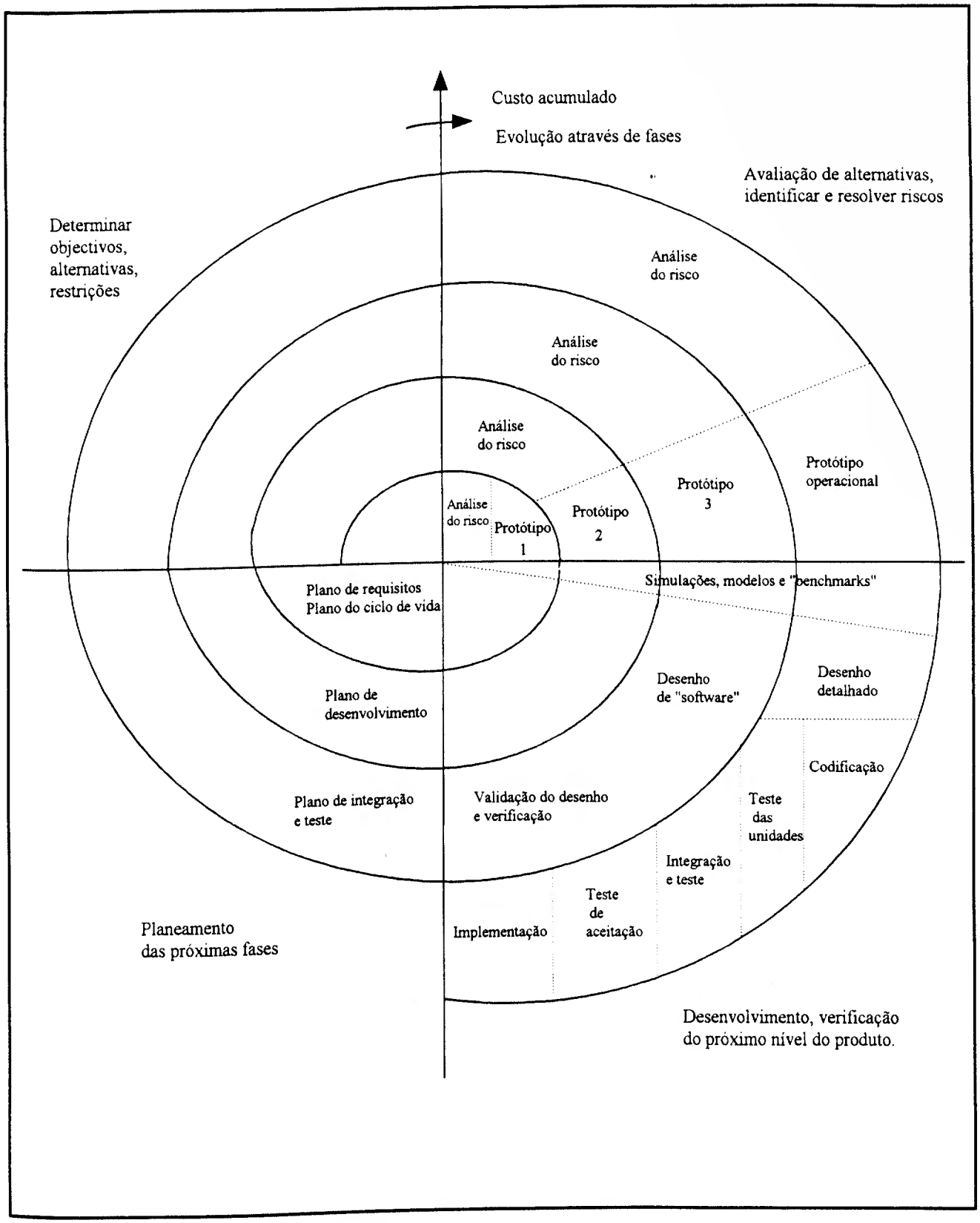


Fig. 2.4 - O modelo em espiral de desenvolvimento de sistemas

FONTE: Boehm, B., « A Spiral Model of Software Development and Enhancement», (Boehm, 1988, p.64)

2.5 Principais problemas com o desenvolvimento de sistemas informáticos

2.5.1 O contexto sócio-económico.

Vivemos num mundo em permanente mudança. A evolução técnica, científica, política, económica e social, permite o aparecimento de novas oportunidades e ameaças que exigem uma resposta adequada por parte das organizações e implicam o reajustamento do seu sistema de informação para fazer face à nova realidade. As empresas tem de estar dotadas de mecanismos que possibilitem uma rápida adaptação a novas situações ambientais.

A superprodução é um problema e reflecte-se na luta constante por quotas de mercado entre produtos que têm de obrigatoriamente ser competitivos.

A internacionalização é uma realidade. As fronteiras comerciais são cada vez mais ténues. O mundo funciona como um palco ao alcance de todos, onde a competição é enorme na procura de oportunidades de negócio. As grandes empresas descentralizam-se, deslocam os seus centros de produção para países onde a mão de obra é mais barata, colocam os seus lucros nos conhecidos "paraísos fiscais", investem consoante a oportunidade sem constrangimentos geográficos, procurando desta forma uma maior rendibilidade dos seus activos.

A estrutura hierárquica clássica com controlo centralizado está rapidamente a dar lugar a uma distribuição e descentralização de poderes, para ser possível responder adequadamente e em tempo oportuno às alterações ambientais (Taylor, 1992a).

A oferta de produtos especializa-se para satisfazer os requisitos e necessidades do mercado. A qualidade é, em muitas indústrias, um factor determinante de sucesso e está na origem, por exemplo, do ascendente da indústria japonesa em termos mundiais.

As tecnologias de suporte aos sistemas de informação sofrem a influência do ambiente sócio-económico e, principalmente, devem permitir às organizações a sua inserção e enquadramento nesse ambiente. A evolução destas tecnologias altera a estrutura das indústrias e as regras pelas quais se rege a competição, cria vantagens competitivas e é responsável pelo nascimento de novas áreas de negócio (Porter e Millar, 1985). Acompanhar o seu processo de evolução é, em muitas indústrias, vital para o sucesso empresarial, pois possibilita a diferenciação dos produtos e a redução de custos.

A complexidade é um vector marcante na sociedade moderna e reflecte-se, nas tecnologias da informação, através da necessidade crescente de sistemas que incorporem e interliguem novos tipos de dados, que possibilitem o armazenamento e processamento de som, fotografia, sequências de video, texto, gráficos, etc.

O desenvolvimento económico e social gera um aumento do volume de dados e transacções, implicando a concepção e desenvolvimento de sistemas informáticos mais rápidos, simultaneamente mais complexos e, sobretudo, flexíveis, de forma a absorverem as alterações ambientais repercutidas no sistema de informação organizacional.

2.5.2 A indústria de *hardware*.

A evolução do *hardware* tem-se processado a um ritmo verdadeiramente alucinante. A descida dos preços e o aumento da capacidade de resposta dos equipamentos é uma constante.

A indústria de *hardware* é caracterizada pela existência de um conjunto de empresas especializadas no aperfeiçoamento e produção de componentes estandardizados (discos, processadores, etc...), cujo custo unitário é relativamente baixo, devido às enormes economias de escala que se obtêm com a sua produção massiva. Estes componentes possuem um elevado grau de adaptação a equipamentos de marcas diferentes.

Os principais "fabricantes" responsáveis pela "marca", apesar dos seus investimentos em investigação e desenvolvimento, concentram-se cada vez mais na montagem de componentes, assim como na comercialização e prestação de serviços de assistência pós-venda.

As redes de computadores e as arquitecturas descentralizadas vêm ao encontro da globalização do comércio internacional, facilitam as comunicações e reflectem, simultaneamente, a descentralização e interligação dos centros de decisão.

2.5.3 A indústria de *software*.

Existe uma notória diferença entre a evolução das tecnologias físicas de suporte aos sistemas de informação e o estádio de desenvolvimento da sua complementar de índole lógico. Este facto limita, substancialmente, o sucesso na *implementação* de sistemas informação.

O processo actual de desenvolvimento de *software* apresenta problemas evidentes, por vezes interligados, que vamos procurar descrever.

2.5.3.1 Custo das aplicações.

O custo das aplicações é normalmente superior ao planeado (Dos Santos *et al*, 1986) e situa-se muitas vezes acima do que seria considerado razoável em termos de custo/benefício.

Comparativamente ao *hardware*, o custo do *software* permanece extraordinariamente alto. No início da década de 80, o custo do *hardware* representava apenas cerca de 20% do valor despendido em *software* (Boehm, 1981).

O elevado custo é um sintoma da deficiente produtividade da indústria de construção de *software*.

2.5.3.2 Produtividade.

Um dos maiores problemas inerentes à informatização das organizações é a evidente falta de produtividade no desenvolvimento de aplicações.

Existe, na grande maioria das empresas (ou departamentos) de desenvolvimento de *software*, uma assinalável procura reprimida ("backlog") de aplicações. Essa procura reprimida pode ser classificada em três tipos diferentes: visível, invisível ou desconhecida (Yourdon, 1989).

A procura reprimida visível envolve os projectos que, apesar de aceites, estão em lista de espera, devido ao facto de não existirem recursos disponíveis para a sua realização. A procura reprimida invisível corresponde aos sistemas que não são solicitados, porque os clientes ou utilizadores têm um número significativo de aplicações cujos pedidos ainda não foram satisfeitos. Na procura reprimida desconhecida incluem-se os sistemas que os utilizadores de momento não sabem que precisam, mas cuja necessidade irá emergir logo que a organização utilize as aplicações classificadas em procura reprimida visível e invisível.

Investigadores do MIT estimaram, em 1982, que a procura reprimida invisível seria cerca de 5,35 vezes maior que a procura reprimida visível [13], o que expressa bem a necessidade urgente de acelerar o processo de desenvolvimento.

---

[13] ALLOWAY, Robert e QUILLARD, Judith, (1982), « User Managers Systems Needs », *CIRS Working Paper 86*, MIT Sloan School for Information Systems Research, Cambridge (Massachussets), citado em YOURDON, Edward, (1989), *Modern Structured Analysis*, 3ª edição, Englewood Cliffs, Prentice Hall, p. 107.

Do ponto de vista metodológico, o processo de desenvolvimento de aplicações sofreu alterações pouco significativas durante as últimas décadas. Apesar de se assinalar uma revolução nas linguagens e "ferramentas" de programação, principalmente com o aparecimento das linguagens de programação de 4ª geração e produtos I-CASE[14], a verdade é que o método de produção ainda é "artesanal". A indústria de *software* necessita de uma "revolução industrial", de forma a passar da fase de "artesanato", onde se procede a uma codificação minuciosa das especificações iniciais, para uma fase de montagem de módulos previamente elaborados e testados, sofrendo apenas, quando necessário, pequenos ajustamentos em função das características específicas de cada aplicação.

O processo de construção de *software* através da reutilização de código permite acréscimos significativos de produtividade e qualidade, paralelamente a uma diminuição no custo de desenvolvimento como resultado da redução do tempo de codificação.

A reutilização do código vem sendo progressivamente realizada utilizando as linguagens de programação estruturada, mas os índices de reutilização não atingiram ainda níveis verdadeiramente satisfatórios. Apesar de se incluírem nas razões apontadas como justificativas deste facto vários aspectos de carácter não técnico [15], como, por exemplo, a relutância dos programadores em incorporarem nas suas aplicações código que não foi desenvolvido na organização, concordamos com Bertrand Meyer quando afirma que " a reutilização de código é limitada porque é difícil desenhar *software* reutilizável" (Meyer, 1988, p.202).

Apesar de aparentemente os programadores escreverem consecutivamente rotinas semelhantes para diferentes aplicações, constata-se que existem, entre essas rotinas, diferenças de detalhe que se revelam significativas. A construção de código reutilizável requer um elevado nível abstracção, paralelamente a uma boa técnica de programação, porque as variáveis a parametrizar, assim como a combinação das diferentes situações de utilização, poderão ser numerosas.

Segundo Meyer, existem boas "bibliotecas" de subrotinas para aplicações numéricas, mas esta técnica apresenta alguns problemas, principalmente com estruturas de dados complexas, porque estas exigem uma distribuição dos dados por diversas subrotinas, eliminando a autonomia na concepção de cada um dos módulos (Meyer, 1988, p.203).

---

[14] Um I-CASE ("Integrated CASE") é uma ferramenta CASE que suporta todas as fases do ciclo de vida do desenvolvimento de sistemas, incluindo a geração automática de código.

[15] Uma descrição exaustiva das causas de não reutilização de código pode ser encontrada em TRACS, Will, « Software Reuse: Motivators and Inhibitors », *in* TRACS, Will, (1988), *Software Reuse: Emerging Technology*, Washington, The Computer Society Press of the IEEE, pp. 62 a 66.

Pensamos que as linguagens de programação estruturada não são, por natureza, linguagens que permitem uma perfeita adopção do conceito de reutilização de software. Segundo Martin e Odell (1992, p.37), dificilmente se poderão atingir os níveis de reutilização de 80% a 90% conseguidos com os métodos de análise, desenho e programação orientada para objectos.

2.5.3.3 Qualidade

A qualidade do *software* está muito longe de ser a desejável. Os sistemas existentes apresentam um elevado índice de erros, apesar do tempo global despendido em testes e depuração de erros atingir normalmente cerca de 50% do tempo de projecto (Yourdon, 1989, p.112).

Os níveis de erro apontados às aplicações são diversos, pois dependem de uma série de factores que se alteram significativamente de projecto para projecto, como, por exemplo, a complexidade do programa, a experiência dos programadores, ou o nível de profundidade com que foi realizado o trabalho de análise.

De qualquer forma, os resultados obtidos através de testes permitem concluir que o nível de *fiabilidade* dos sistemas é insuficiente. Num estudo efectuado pela Hewlett-Packard, em 57 dos seus projectos, o índice médio de defeitos encontrados (em programas com um nível de reutilização de código inferior a 75%) foi de 8,12 defeitos por cada 1000 linhas de código (Grady e Caswell, 1987, p.239). Alguns outros testes publicados transmitem resultados mais pessimistas. Num estudo efectuado por Carpers Jones, o autor apresenta níveis de erros entre 49,5 a 94,6 (erros por 1000 linhas de código) para cinco projectos de diferentes tipos e dimensões (Jones, 1986, p.172).

São conhecidos inúmeros exemplos de falhas de sistemas informáticos[16], acrescido do facto de existirem erros que simplesmente não são detectados, e outros que não são divulgados para não desprestigiar a imagem pública da organização. Esta situação torna-se preocupante se atendermos à importância e responsabilidade atribuída a alguns desses sistemas informáticos.

---

[16] Alguns casos são apresentados em NEUMANN, Peter G., (Janeiro 1985), « Some Computer-Related Disasters and Other Egregious Horrors », *ACM SIGSOFT Software Engineering Notes.*

2.5.3.4 Manutenção.

Entendemos por manutenção as alterações que têm de ser efectuadas no sistema informático após este estar em funcionamento.

Segundo Boehm (1981), o custo de manutenção é normalmente superior ao custo de desenvolvimento. Vários são os estudos que abordam esta questão e as diferentes estimativas apontam para que 40% a 80% dos recursos informáticos estejam afectos à manutenção (Martin, 1983, p.23). Citando Amílcar Sernadas: "sabemos que, hoje como ontem, 80% dos recursos humanos da empresa típica estão dedicados à manutenção dos sistemas existentes e apenas 20% estão livres para responder à necessidade de novas aplicações" (Sernadas, 1993, p.4).

Swanson (1976) distingue três tipos de manutenção, que designa de correctiva, adaptativa e perfectiva.

A manutenção correctiva corresponde à identificação e correcção de erros existentes no *software*, erros esses que poderão ter origem nas diversas fases do ciclo de desenvolvimento.

A manutenção adaptativa relaciona-se com as modificações que é necessário efectuar por motivos operacionais (como por exemplo, aumentar o número de caracteres de um determinado código), ou aquelas que derivam de alterações no sistema de suporte das aplicações (como sejam, a aquisição de um novo equipamento ou a utilização de um sistema operativo diferente).

Entende-se por manutenção perfectiva, as alterações a realizar em virtude de novos requisitos decorrentes da evolução do ambiente organizacional, ou outras que visem aumentar a eficiência ou facilitar a manutenção das aplicações.

Sobre estes três tipos de manutenção julgamos relevante assinalar alguns aspectos apontados por Martin e McClure (1983):

- A correcção de erros constitui apenas 20% do esforço de manutenção e o seu peso tende a diminuir com a utilização de novos métodos e instrumentos de desenvolvimento de aplicações.
- A manutenção adaptativa é estimada em 25%. Prevê-se a estabilidade deste valor devido à necessidade permanente de ajustamento das aplicações às novas tecnologias constantemente emergentes.
- A parte mais significativa do esforço de manutenção (55%) concentra-se no contínuo aperfeiçoamento das aplicações como resultado das alterações do meio envolvente.

Atendendo aos factos anteriormente assinalados, podemos concluir que não existem aplicações completamente terminadas ou acabadas, subsistindo uma necessidade contínua e profícua de aperfeiçoamento das aplicações existentes com o objectivo de as adaptar às alterações provenientes do contexto organizacional.

A manutenção será sempre um mal necessário, com o qual os sistemas informáticos têm de conviver, mas é importante assinalar que, apesar da sua indispensabilidade, o nível de custos e recursos concentrado nesta fase do ciclo de vida afigura-se excessivo, sendo sintoma da deficiente qualidade das aplicações produzidas. Essa deficiente qualidade traduz-se num elevado número de falhas e tem origem em deficiências metodológicas na elaboração das aplicações que dificultam o respectivo processo de correcção e aperfeiçoamento.

Capítulo III

A tecnologia orientada para objectos

A orientação para objectos constitui um novo paradigma, na medida em que representa uma diferente e extensiva forma de interpretação da realidade, com conceitos próprios, colocando em causa os princípios e modelos anteriormente definidos. Apesar do seu impacto se revelar no desenvolvimento de sistemas informáticos, concordamos com Thomsett (1990) quando afirma que devemos olhar para o paradigma do desenvolvimento orientado para objectos, primeiro, como um paradigma organizacional e, em segundo lugar, como um paradigma técnico.

É objectivo deste capítulo apresentar os principais conceitos da tecnologia orientada para objectos, principalmente aqueles com maior expressão na análise de sistemas de informação, assim como assinalar o estado actual desta tecnologia.

3.1 Breve perspectiva histórica.

A tecnologia orientada para objectos teve o seu início nos anos 60 e o seu primeiro marco surgiu através do aparecimento da linguagem de programação Simula 67, desenvolvida na Noruega por Ole-Johan Dahl e Kristen Nygaard[1] . O Simula 67, derivado a partir do Algol 60, era fundamentalmente uma linguagem de programação de simulações de processos industriais em computador e possuía os conceitos de classe e objecto. Os princípios conceptuais desta linguagem marcaram significativamente as posteriores linguagens de programação integralmente orientadas para objectos, como o Smalltalk [2] ou o Eiffel [3] .

---

[1] DAHL, O.-J. e Nygaard, K., (1966), « Simula, an ALGOL based simulation language », *Communications of the ACM*, vol. 9, nº 9, pp. 671-678 e DAHL, O. J., MYHRHAUG, B. e NYGAARD, K., (1970), « The Simula 67 Common Base Language », Publication S22, Oslo, Norwegian Computing Centre.

[2] Apesar de ser inicialmente desenvolvido entre 1970 e 1972 (Smalltalk 72), por um grupo de investigadores da Xerox em Palo Alto, liderado por Alan Key, o Smalltalk só mais tarde, em 1980, foi comercializado numa nova versão: Smalltalk 80. Uma descrição da linguagem pode ser encontrada em GOLDBERG, A. e ROBSON, D., (1983), *Smalltalk 80: The language and it's implementation*, Reading, Addison-Wesley.

Durante a década de 80, diversas foram as linguagens de programação estruturadas que adoptaram uma filosofia de orientação para objectos. Alguns exemplos são o Object-Pascal (definido por Tesler e Wirth, em 1984), o Objective C (Cox, 1986) e o C++ (Stroustrup, 1986) [4].

O campo de aplicação da tecnologia orientada para objectos estende-se a todo o ciclo de vida de desenvolvimento de *software*. Para a sua correcta implantação é importante o desenvolvimento de métodos de análise e desenho, de produtos CASE, e também de SGDBs (Sistemas de Gestão de Bases de Dados) que incorporem esta filosofia.

A era dos Sistemas de Gestão de Bases de Dados orientados para objectos surgiu fundamentalmente a partir da segunda metade dos anos 80. Apesar de actualmente existirem mais de uma dezena de Sistemas de Gestão de Bases de Dados considerados como orientados para objectos[5], a sua difusão é relativamente diminuta, restringindo-se basicamente a projectos de investigação científica.

A definição de SGBD orientado para objectos é motivo de alguma polémica. Apesar de alguns esforços desenvolvidos, como, por exemplo, a elaboração por um grupo de reconhecidos investigadores do "Object-Oriented Database Manifesto" [6], não existe para os SGBDs orientados para objectos uma definição indiscutível e geralmente aceite pela comunidade científica, como acontece com os sistemas relacionais, pormenorizadamente definidos por E. Codd em 1985 [7].

---

[3] A linguagem é descrita em MEYER, Bertrand, (1988), *Object-Oriented Software Construction*, Prentice-Hall, Hemel Hempstead; e posteriormente desenvolvida em MEYER, Bertrand, (1991), *Eiffel: The language*, New York, Prentice-Hall.

[4] Uma descrição da evolução das linguagens de programação orientadas para objectos é apresentada nas seguintes obras:

HENDERSON-SELLERS, Brian, (1992), *A Book of Object-Oriented Knowledge*, Englewood Cliffs, Prentice Hall;

KHOSHAFIAN, Strag e ABNOUS, Razmik, (1990), *Object-Orientation: Concepts, Languages, Databases, User Interfaces*, New York, John Wiley & Sons, pp. 11 a 30;

SIGS Publications, (1992), « Special Silver Anniversary Supplement », Suplemento da edição de *C++ Report*, vol. 4, nº 9.

[5] Em BARRY, John, (1992), « The World of Object-Oriented DBMSs », *DBMS Buyers Guide*, vol. 5 nº 7, pp. 53 a 55, identificamos 14 Sistemas de Gestão de Bases de Dados orientados para objectos, comercialmente disponíveis nos Estados Unidos.

[6] O "Object-Oriented Database Manifesto", inclui 13 regras que procuram definir uma base de dados orientada para objectos. Este documento foi escrito em 1989 por Malcolm Atkinson, Francois Bancilhon, David DeWitt, Klaus Kittrick, David Maier e Stanley Zdonik (Edelstein, 1991).

[7] CODD, E. F., (Outubro 1985), « Is Your DBMS Really Relational », *Computer World*.

De qualquer modo, podemos dizer que os SGBDs orientados para objectos incorporam os conceitos inerentes ao paradigma de orientação para objectos, conjuntamente com as características tradicionais dos sistemas de gestão de bases de dados (persistência, controlo de transacções, mecanismos de recuperação de dados, linguagens de interrogação, integridade, segurança, etc).

Os SGBDs orientados para objectos estão especialmente vocacionados para interrogar e manipular complexas estruturas de dados, permitindo, neste caso, maior rapidez de acesso do que os SGBDs reticulados, hierárquicos ou relacionais [8]. Entendemos por "complexa estrutura de dados" uma estrutura onde poderão existir milhares de diferentes tipos de entidades ou tabelas com elevado número de relações existentes entre si (Edelstein, 1991). Paralelamente, é reconhecido que os SGBDs orientados para objectos apresentam algumas desvantagens no processamento de simples transacções (Taylor, 1992a, p.178).

Estão igualmente disponíveis no mercado alguns produtos OOCASE [9], embora as suas potencialidades de momento sejam relativamente limitadas. É perfeitamente natural o estado de subdesenvolvimento das tecnologias OOCASE, pois, para a sua concretização, é necessário um método de suporte que inclua pelo menos as fases de análise e desenho de sistemas, e estes métodos só muito recentemente apresentam alguma consistência. Por outro lado, a adopção de novas formas de desenvolvimento através da utilização de ferramentas CASE envolve custos de reformulação de recursos e reflecte-se na própria cultura organizacional, verificando-se normalmente uma considerável resistência à mudança [10].

---

[8] Em CATTELL, Roderic G, (1991), *Object Data Management*, Reading, Addison-Wesley, podemos constatar os resultados proporcionados por vários testes efectuados pela SUN Microsystems Inc., com o intuito de avaliar a capacidade de resposta dos Sistemas de Gestão de Bases de Dados orientados para objectos, em comparação com os sistemas relacionais, na manipulação de complexas estruturas de dados. Em alguns testes de pesquisa, que não exigem operações de actualização de dados em disco, os sistemas orientados para objectos chegaram a resultados cerca de 100 vezes mais rápidos que os relacionais (Cattell, 1991, p.201).

[9] *Object-Oriented Computer Aided Software Engineering*. Uma descrição sumária dos produtos orientados para objectos actualmente existentes pode ser encontrada em: SALMONS, Jim e BABITSKI, Timlynn, 1992, *1992 International OOP Directory*, New York, SIGS Publications.

[10] MERLYN, Vauhghan, comunicação intitulada "A Gestão dos Impactos Organizacionais originados pela introdução de CASE ", proferida no seminário "CASE - Que evoluções ? Que atitudes ?", Lisboa, Outubro de 1992.

Na segunda metade da década de 80 surgem as primeiras abordagens sobre análise e desenho orientado para objectos, mas os métodos actualmente mais utilizados e divulgados foram publicados, com um nível satisfatório de detalhe, principalmente na década de 90 [11].

3.2 Conceitos de base.

O paradigma da orientação para objectos incorpora um conjunto de conceitos que são fundamentais para a sua compreensão. Existem algumas diferenças de perspectiva na abordagem destes conceitos pelos diversos autores, resultado simultâneo do relativo estado de imaturidade da respectiva tecnologia e da grande amplitude do seu campo de aplicação, que se estende ao longo do ciclo de vida do desenvolvimento de *software*.

3.2.1 O conceito de objecto.

Não existe uma noção normalizada do conceito de objecto. A sua definição deriva usualmente do contexto da sua utilização.

Numa perspectiva conivente com a análise de sistemas de informação, Martin e Odell entendem por objecto: "qualquer coisa real ou abstracta, sobre a qual guardamos dados e métodos que manipulam esses dados" (Martin e Odell, 1992, p.16).

Um método corresponde a um procedimento que estipula a forma como os dados são manipulados. Enquanto as propriedades do objecto são expressas pelo valor dos respectivos atributos, os métodos definem as operações que ele pode efectuar.

A anterior definição de objecto, apesar de abordar a essência do conceito (que consiste na identificação de "entidades" que incorporam paralelamente dados e métodos), é relativamente restrita e não exprime toda a riqueza e complexidade inerente à noção de objecto.

---

[11] A título de exemplo podemos citar, entre outras, as seguintes obras:
COAD, Peter e YOURDON, (1990), Edward, *Object-Oriented Analysis*, Englewood Cliffs, Prentice Hall.
RUMBAUGH, James e al., (1991), *Object-Oriented Modeling and Design*, Englewood Cliffs, Prentice Hall, (apresenta pormenorizadamente o método OMT - Object Modeling Technique).
BOOCH, Grady, (1991), *Object-Oriented Design with applications*, Redwood City, Benjamin/Cummings.
MARTIN, James e ODELL, James, (1992), *Object-Oriented Analysis and Design*, Englewood Cliffs, Prentice Hall.

Uma definição mais completa, e simultaneamente mais complexa, é adoptada pelo OBLOG [12], considerando um objecto como "uma entidade passiva ou activa, mais ou menos persistente, com atributos dependentes do estado, com acções capazes de alterar esse estado, capaz de viver concorrentemente com outros objectos, com os quais pode interactuar, quer por partilha de atributos (partilha de memória), quer por partilha de eventos (troca de mensagens), capaz de se agregar com outros objectos, em objectos compostos, e capaz de herdar características de outro objecto." (Sernadas *et al.*, 1989a, p.1.19).

Entende-se, no contexto do OBLOG, por entidades passivas, as mensagens recebidas pelo sistema e respectivos registos, e por entidades activas os processamentos efectuados sobre as mensagens que o sistema recebeu e memorizou.

De uma forma sumária, podemos dizer que um objecto é uma realidade que possui identidade, estado, estrutura e comportamento.

A identidade de um objecto permanece imutável ao longo da sua vida e permite distingui-lo dos restantes. Através do conceito de identidade, os objectos podem incorporar ou referenciar outros objectos.

Durante o seu tempo de vida, um objecto pode estar em diferentes estados, adoptando o comportamento e propriedades relativas a cada estado. Por exemplo, o objecto cidadão, pode assimilar os estados de aluno, militar, funcionário público e reformado.

Os estados serão activados mediante a realização de eventos. Um evento é algo que ocorre num determinado instante no tempo e que poderá afectar o estado dos objectos. O facto de um funcionário atingir uma determinada idade permite a sua passagem ao estado de reformado, adquirindo atributos (como, por exemplo, uma pensão de determinado valor) e comportamentos (necessidade de efectuar periodicamente uma prova de vida) próprios desse novo estado.

3.2.2 Abstracção.

A noção de abstracção é um dos suportes do paradigma da orientação para objectos, pois representa um poderoso mecanismo para lidar com a diversificação e a complexidade existente no mundo real. A abstracção consiste numa função mental que permite visualizar a realidade com vários níveis de detalhe.

---

[12] O OBLOG (OBject-oriented LOGic), desenvolvido no INESC a partir de 1986, corresponde a uma abordagem orientada para objectos que incorpora um conjunto de conceitos, técnicas e linguagens, que possibilitam a especificação formal de sistemas de informação.

O processo de abstracção é psicologicamente necessário e natural (Wirfs-Brock *et al.*, 1990, p.3), sendo crucial para o entendimento do mundo que nos rodeia. Atribuindo ênfase a algumas semelhanças e desprezando diferenças, a abstracção possibilita a classificação dos objectos em diferentes classes, de acordo com o interesse e perspectiva psicológica do indivíduo que capta a informação proveniente do real.

O mecanismo intelectual de abstracção é inato (apesar de naturalmente poder ser desenvolvido através do treino) e está subjacente à concepção de qualquer modelo de análise de sistemas de informação, quer ele seja construído com base numa perspectiva estruturada ou orientada para objectos.

Apesar de, na análise de sistemas de informação orientada para objectos, a aplicação do mecanismo de abstracção não ser inovador, ele é utilizado com maior intensidade. A explicação reside no facto de o grau de complexidade inerente ao conceito de "objecto" ser significativamente superior ao grau de complexidade do conceito de "entidade" (tal como este é entendido na análise estruturada) e das estruturas de um modelo orientado para objectos envolverem uma maior hierarquização dos seus componentes.

3.2.3 Tipos abstractos de dados[13] e classes.

Um tipo (ou tipo abstracto de dados) é definido por uma estrutura de atributos, mensagens e métodos, que poderão ser invocados pelos objectos. Os tipos abstractos de dados são identificados durante a fase de análise de sistemas de informação.

A diferença fundamental entre o conceito de entidade, tal como ele é entendido no modelo "entidade-associação" (Chen, 1976), e o conceito de tipo, é que, enquanto a entidade se expressa através de uma estrutura de dados, um tipo define não só a estrutura de dados como inclui os métodos que manipulam esses dados.

Uma classe corresponde à *implementação* de um tipo (Henderson-Sellers, 1992, p.229). Os detalhes referentes aos métodos, identificados no tipo, e aplicáveis sobre os objectos, são especificados na classe.

No contexto das linguagens de programação orientadas para objectos, enquanto que o tipo é usualmente referenciado como uma noção de tempo de compilação ("compile time"), com importância na verificação da correcção dos programas, o conceito de classe é um conceito de tempo de execução ("run-time") que incorpora detalhes de *implementação* e

---

[13] A designação de "tipo abstracto de dados" corresponde à tradução para português de "abstract data type".

associa um tipo a um conjunto de objectos, designados de instâncias da classe (Velho, 1991, p. 19).

Os conceitos de tipo e de classe apresentam evidentes semelhanças entre si. Alguns autores não estabelecem uma distinção clara entre os termos e são várias as linguagens de programação que utilizam apenas um destes conceitos [14].

Por motivos de expressividade semântica, existe, por vezes, a necessidade de definir classes no topo de uma estrutura hierárquica, desprovidas de instâncias. Essas classes designam-se de classes abstractas. As classes abstractas são criadas recorrendo ao processo de abstracção, o qual permite realçar processos e atributos pertencentes aos objectos que estão incorporados nas classes concretas, hierarquicamente dependentes das classes abstractas. Designa-se de classe concreta uma classe não abstracta que possui, portanto, instâncias.

3.2.4 Hereditariedade

Uma classe pode herdar métodos e atributos de outra classe. Sendo assim, uma classe de objectos pode ser definida como um caso particular de uma classe mais global, automaticamente incluindo os métodos e atributos definidos nessa outra classe.

Se uma determinada classe herda características estruturais e comportamentos de outra, a primeira designa-se de subclasse e a segunda de superclasse. A herança directa de atributos e métodos de mais do que uma classe, designa-se de múltipla hereditariedade. Além dos métodos e atributos herdados, as subclasses dispõe dos seus próprios métodos e atributos.

3.2.5 Mecanismos de relação entre classes.

Os modelos semânticos de dados, desenvolvidos durante a década de 70, possuem mecanismos de relação que vão ser adoptados pelos modelos orientados para objectos, nomeadamente: associação, agregação e generalização.

---

[14] O Smalltalk 80, por exemplo, apenas utiliza a noção de classe (Goldberg *et al.*, 1983).

Uma relação entre classes pode expressar a conveniência na partilha de métodos e atributos, ou indicar uma interligação de carácter semântico entre as classes (Booch, 1991, p.96).

A associação[15] é um mecanismo que permite relacionar classes que, apesar de diferentes na sua essência, possuem alguma correspondência entre si. Por exemplo, automóvel e condutor, apesar de possuírem diferentes características, interagem no desempenho da sua missão, pelo que será eventualmente relevante dispor de informação sobre estas duas classes conjuntamente relacionadas.

A generalização corresponde a uma forma de abstracção em que, realçando-se as suas semelhanças e suprimindo-se as respectivas diferenças, objectos similares são relacionados com um objecto genérico. Numa relação de generalização, a classe que define os aspectos comuns aos objectos, designa-se de genérica, e as classes que contêm características específicas, não comuns entre si, denominam-se de especializadas.

Uma generalização poderá ocorrer por motivos de extensão, onde a classe especializada acrescenta atributos e métodos particulares aos métodos e atributos existentes na classe genérica, ou com o objectivo de introduzir restrições para as características definidas na classe genérica (Rumbaugh *et al.*, 1991).

As estruturas de generalização permitem a concretização do conceito de hereditariedade. Uma classe C é uma generalização das classes C1...Cn, se cada objecto de C é também um objecto de uma das classes C1...Cn. A generalização é normalmente exclusiva, o que significa que os conjuntos de objectos das classes especializadas são disjuntos. Com a introdução do conceito de múltipla hereditariedade, existe subjacente a possibilidade de um objecto pertencer, simultaneamente, a duas classes posicionadas no mesmo nível de uma estrutura hierárquica de generalização. As classes veículo terrestre e veículo aquático, por exemplo, são especializações de uma classe genérica veículo. Um veículo anfíbio, na medida em possui características das duas classes anteriormente assinaladas, exemplifica uma situação de múltipla hereditariedade.

A agregação é uma forma de relação em que o objecto agregado é constituído por partes ou componentes (Rumbaugh *et al.*, 1991) [16]. Os componentes são representados por

---

[15] Corresponde ao conceito de "relationship" do modelo "Entity-Relationship" ("Entidade-Associação") (Chen, 1976).

[16] A agregação, aqui definida, corresponde a uma adaptação para a modelação orientada para objectos da estrutura hierárquica usualmente designada de "todo-parte" ("part-whole"), apresentada como extensão ao modelo entidade-associação (Teorey, 1986). O conceito de agregação inicialmente expresso em Smith e Smith (1977) é significativamente diferente do adaptado por nós e utilizado por Rumbaugh (1991).

outros objectos que podem, ou não, existir separados do agregado. Um automóvel é constituído por diversas partes distintas, como sejam, o motor, pneus, *chassis*, etc. Estes componentes correspondem a objectos, se a sua relevância no universo em análise justificar a identificação e concepção das respectivas classes.

A agregação é uma relação transitiva. Se C1 é um agregado de C2, e C2 é um agregado de C3, então C1 também é um agregado de C3.

A agregação de objectos pode ser considerada fixa, variável ou recursiva (Rumbaugh *et al.*, 1991). Uma agregação fixa é caracterizada por um número estável de níveis de decomposição e de componentes em cada nível. Numa agregação variável existe um número fixo de níveis de decomposição, mas o número de componentes referenciados em cada nível pode variar. Uma agregação recursiva é um caso particular de agregação em que um componente contêm agregado outro componente do mesmo tipo.

A estruturação dos diferentes elementos do mundo real, numa perspectiva de orientação para objectos, é realizada recorrendo frequentemente à hierarquização desses elementos. Uma hierarquia corresponde a uma classificação ordenada de abstracções elaborada de forma a facilitar a compreensão da realidade (Booch, 1991, p.54). A hierarquização concretiza-se recorrendo aos mecanismos de relação de agregação e generalização.

3.2.6 Capsulação, polimorfismo e comunicação entre objectos.

Entende-se por capsulação[17] o facto de dados e métodos serem conjuntamente incorporados, permitindo esconder dos utilizadores os detalhes de construção do objecto.

A comunicação entre objectos realiza-se através do envio de mensagens que permitem activar os métodos (previamente definidos) de uma determinada classe. Uma mensagem é constituída pelo nome do objecto, o nome do método a evocar e um grupo de parâmetros que caracterizam a forma como o método irá operar.

Se uma mensagem é enviada para uma determinada classe, não se conhecendo qual a classe que fisicamente incorpora o método que permite responder a esse estímulo, diz-se que

---

[17] O termo "capsulação" corresponde à nossa tradução para português do vocábulo inglês "encapsulation". É vulgar, em livros e artigos escritos ou transcritos para língua portuguesa, a utilização do termo "encapsulação" ou "encapsulamento" (este último usual em edições brasileiras). Esta tradução é em nosso entender incorrecta, primeiro, porque tais termos não existem no dicionário da língua portuguesa (COSTA, J., e MELO, A, (1991), Dicionário da Língua Portuguesa, 6ª edição, Porto, Porto Editora), segundo, porque existe um outro termo ("capsulação") que está definido com um significado idêntico a "encapsulation".

estamos perante uma situação de polimorfismo. Quando uma subclasse recebe uma mensagem, para execução de uma determinada operação, efectua-se nessa subclasse uma pesquisa do método correspondente, e caso esse método não seja detectado será examinada a respectiva superclasse, e assim sucessivamente. O método a executar poderá ser herdado de uma outra classe que não aquela para a qual foi inicialmente dirigida a mensagem. O conceito de polimorfismo representa o facto de objectos de diferentes classes poderem responder de uma forma própria e distinta a uma mensagem comum.

Um processo de desenvolvimento de sistemas puramente orientado para objectos baseia-se numa filosofia em que qualquer realidade informacional pode ser considerada um objecto. Os sistemas informáticos, do ponto de vista lógico, devem representar a simulação do intercâmbio de mensagens entre os objectos, como se estes fossem dotados de vida própria. Os detalhes relativos à estrutura de dados e processos das diferentes classes não são publicamente conhecidos. A classe comunica com o exterior através do seu *interface*, o qual permite a recepção de mensagens que possam ser interpretadas, originando uma resposta.

Para Grady Booch, qualquer que seja o seu campo de aplicação (programação, desenho ou análise), a estrutura conceptual que define uma perspectiva de orientação para objectos é o modelo de objectos (Booch, 1991, p.38). O modelo de objectos baseia-se em quatro conceitos fundamentais: abstracção, capsulação, modularidade e hierarquia.

Enquanto a capsulação ajuda a gerir a complexidade, escondendo os detalhes relativos ao elevado número de classes de objectos criadas a partir da aplicação do processo de abstracção, a modularidade permite mais facilmente a manipulação da complexidade, através da possibilidade de dividir um problema em diferentes componentes individuais. Por sua vez, as estruturas hierárquicas de generalização e agregação desempenham um papel de relevo na estruturação das diferentes classes de objectos.

3.3 O desenvolvimento de sistemas através de uma orientação para objectos.

Booch defende que o desenvolvimento de *software* orientado para objectos deve ser incremental e interactivamente realizado com base no modelo em espiral de Boehm (1988). O respectivo ciclo de vida contempla as fases de análise, desenho, evolução e manutenção (Booch, 1991).

As funções e objectivos das fases de análise e desenho são distintos e devem ser perfeitamente delimitados. Apesar da análise anteceder a fase de desenho, esta última deverá começar antes da primeira estar concluída. Booch sugere uma estratégia de intercâmbio permanente entre as duas fases.

Na análise de sistemas orientada para objectos, "deve-se procurar modelar o mundo através da identificação de classes e objectos que formam o vocabulário do domínio do problema, enquanto no desenho orientado para objectos, se inventam abstracções e mecanismos que proporcionam o comportamento requerido pelo modelo" (Booch, 1991, p.141).

A fase de análise deve permitir o entendimento entre a equipa de desenvolvimento e os utilizadores, através de um vocabulário comum que possibilite exprimir a realidade inerente ao domínio do problema em estudo, proporcionando uma descrição completa, clara e consistente.

Apesar de considerar que a análise orientada para objectos é ideal para a realização do desenho orientado para objectos, pois proporciona a identificação das classes de objectos a especificar e relacionar, Booch admite igualmente que o desenho orientado para objectos pode ser antecedido por uma análise estruturada ou por simples descrições informais do problema. As técnicas estruturadas apresentam como vantagens o facto de serem substancialmente conhecidas, existindo diversas "ferramentas" de desenvolvimento que suportam a sua notação (Booch, 1991, p.201).

A fase de evolução integra as tradicionais fases de codificação, teste e instalação. O processo de desenvolvimento com desenho orientado para objectos baseia-se na "produção incremental de uma série de protótipos que eventualmente conduzirão à implementação final" (Booch, 1991, p.202).

Na fase de modificação procede-se às necessárias alterações para que o sistema concebido permita responder à evolução ocorrida no meio envolvente.

Em nossa opinião, a concepção que Booch possui de análise orientada para objectos é relativamente restrita, limitando-se fundamentalmente à identificação de classes e objectos, procurando concentrar no desenho os aspectos referentes à especificação do funcionamento do sistema de informação a automatizar (e que vão condicionar o sistema computacional a

conceber). É na fase de desenho que se "inventam as abstracções e mecanismos que proporcionam o comportamento que o modelo requer" (Booch, 1991, p.200).

Wirfs-Brock considera que o desenvolvimento orientado para objectos se inicia com a especificação de requisitos, cujo objectivo é "descrever o que é que o *software* deve ou não contemplar" (Wirfs-Brock *et al.*, 1990, p.9), seguindo-se uma fase de desenho cujo resultado final será "um sistema de objectos que preencha os requisitos, uma descrição do comportamento público desses objectos, e padrões de comunicação entre os objectos" (Wirfs-Brock *et al.*, 1990, p.10). As restantes fases identificadas para o ciclo de desenvolvimento são designadas de implementação, teste, manutenção, e por último, refinamento e extensão.

A perspectiva de Wirfs-Brock é profundamente orientada para a implementação, no sentido em que se procura rapidamente desenhar um sistema de computacional, a partir de requisitos iniciais, sem contemplar um dispêndio assinalável de tempo na análise do sistema de informação. Os objectivos atribuídos por Booch à análise orientada para objectos estão supostamente incluídos na fase de desenho de Wirfs-Brock.

Taylor (1993) apresenta uma estrutura de ciclo de vida para desenvolvimento de sistemas de informação organizacionais, através de tecnologia orientada para objectos, contemplando quatro fases:

1. Desenho de um modelo da actividade empresarial.
2. Construção de classes que suportem esse modelo.
3. Montagem do modelo a partir da incorporação de classes.
4. Adicionar interfaces para resolução de problemas empresariais.

A primeira etapa definida por Taylor inclui as fases de análise e desenho do ciclo de vida convencional do desenvolvimento de sistemas.

O modelo em fonte ("fountain model") (Henderson-Sellers e Edwards, 1990), representado graficamente na fig. 3.1, simboliza as principais fases de desenvolvimento através de círculos cujas áreas de intercepção representam a interligação e fusão parcial existente entre as diversas fases. As setas em sentido descendente expressam o processo de realimentação permanente das fases anteriores em função dos resultados obtidos na elaboração dos diversos protótipos. O decréscimo de manutenção, decorrente da utilização de métodos orientados para objectos, é graficamente representado através da redução do diâmetro do círculo referente à fase de manutenção.

Segundo Booch (1991), a análise do tempo e recursos humanos necessários para a realização das diferentes fases permite concluir que o desenho assume uma assinalável preponderância.

Numa comparação com a utilização de métodos estruturados, o desenho é a única fase que, no desenvolvimento orientado para objectos, necessita de maior número de recursos humanos ou, preferencialmente, um número idêntico ou inferior mas afectados durante um maior período de tempo (Booch, 1991, p.206 a 208). 'As vantagens, em termos de redução de recursos, proporcionadas pela utilização de métodos de desenvolvimento orientados para objectos revelam-se nas fases posteriores ao desenho, nomeadamente na codificação, teste, integração (Booch, 1991, p.207) e também na manutenção do sistema informático.

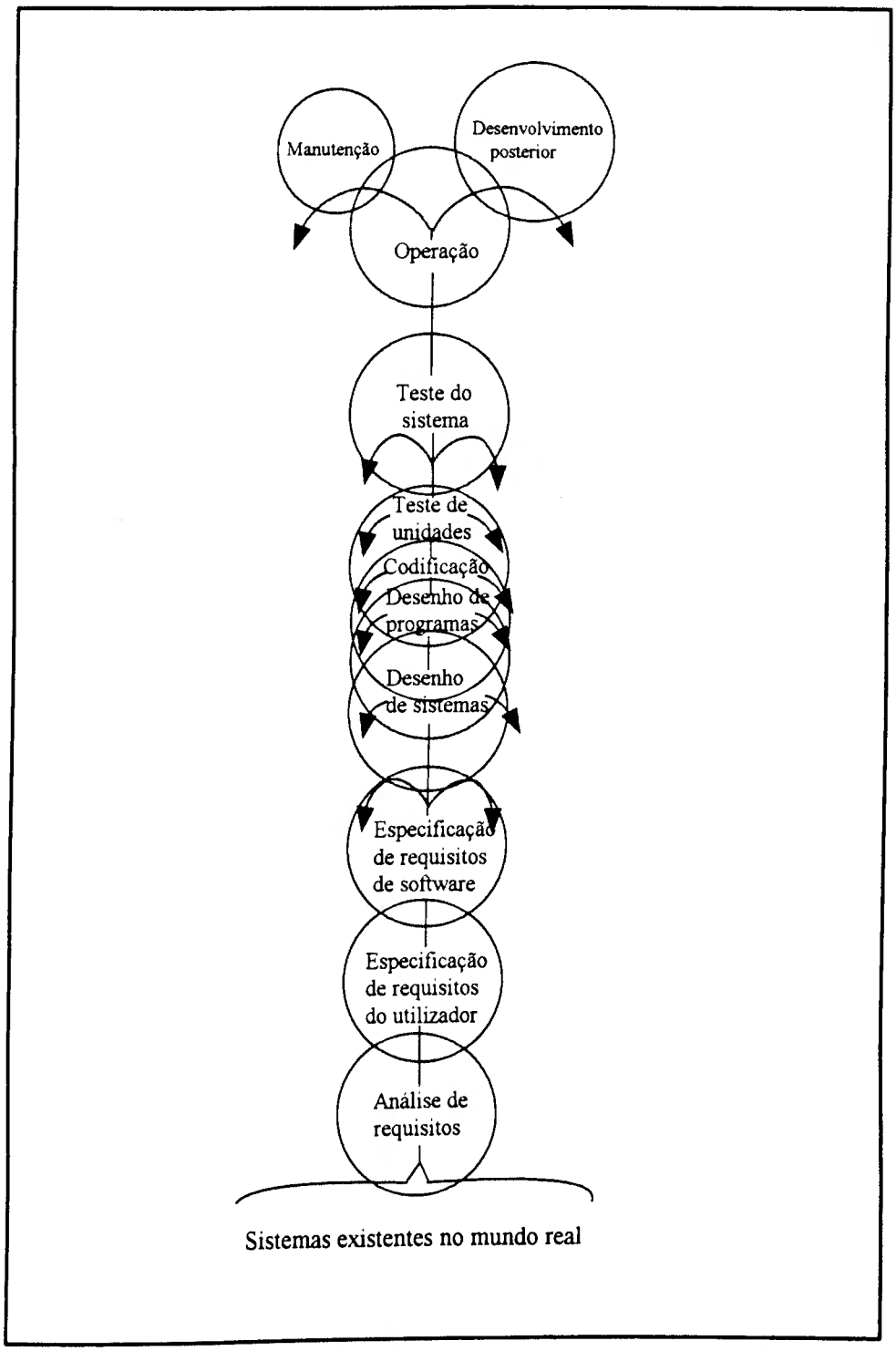


Fig. 3.1 - O modelo em fonte de desenvolvimento de sistemas.

FONTE: Henderson-Sellers, B. e Edwards, J., « The Object-Oriented Systems Life Cycle» (1990, p.152)

3.4 Principais vantagens e problemas com a utilização de tecnologia orientada para objectos.

3.4.1 Vantagens potenciais.

a) Maior expressividade.

A orientação para objectos fundamenta-se numa diferente filosofia de observação e análise da realidade, apresentando uma maior capacidade de representação semântica do sistema em estudo. O modelo relacional (Codd, 1970) força a uma forma tabelar, normalizada, a representação de uma realidade complexa constituída por uma multiplicidade de objectos de diferentes formas. Nos modelos de dados orientados para objectos não existe a necessidade de normalização[18]. O conceito de objecto, conjuntamente com a intensa utilização do processo de abstracção, possibilita a necessária diferenciação, interligação e classificação dos elementos do mundo real segundo as suas características naturais, não só em termos de atributos como também de comportamento.

O paradigma da orientação para objectos não só representa vantagens numa perspectiva técnica de automatização dos sistemas de informação como também apresenta um novo modelo de interpretação e compreensão da realidade. O sistema organizacional é composto por um vasto conjunto de objectos interactuantes que definem, em função da sua acção, a orientação e desempenho da organização.

b) Acelerar o processo de desenvolvimento, melhorando a qualidade do *software*.

A tecnologia orientada para objectos permite acelerar o processo de desenvolvimento de sistemas informáticos com base em três vectores: construção de *software* a partir de objectos estandardizados e "pré-fabricados", maior facilidade de

[18] Entendemos aqui por "normalização" a aplicação das "Formas Normais" definidas em CODD, E.F.,(1970), « A Relational Model of Data for Large Shared Data Banks », *Communications of the ACM*, vol. 13, nº6.

reutilização de código e substituição do processo convencional de desenvolvimento pela rápida construção de protótipos (Taylor, 1991a).

A construção de *software* por montagem e incorporação de classes de objectos previamente definidas permite encurtar significativamente a duração das fases de programação e teste.

O conceito de reutilização tem maior expressão no paradigma da orientação para objectos e a sua adequada utilização revela-se de importância fundamental para a evolução do processo de desenvolvimento de *software*. A capsulação simultânea de dados e processos define o objecto como uma unidade modelarmente completa e independente, com maior capacidade de adaptação e integração num novo contexto organizacional.

A reutilização permite não só acelerar o processo de desenvolvimento como garante uma maior qualidade das aplicações, devido ao facto de o código reutilizado estar previamente testado.

O rápido desenvolvimento de protótipos reduz significativamente o tempo despendido nas fases de análise e desenho de sistemas, assim como flexibiliza a realização de eventuais alterações e extensões às aplicações.

O conceito de hereditariedade traduz-se numa maior facilidade de manutenção do *software*, na medida em que as alterações efectuadas numa classe reflectem-se imediatamente em todas as subclasses de si dependentes. Por outro lado, o incremento na *fiabilidade* das aplicações implica que não seja necessário despender um acentuado esforço na sua manutenção.

c) Diminuição de custos.

A redução da duração das fases de programação e teste, o incremento supostamente obtido na qualidade das aplicações e a facilidade de manutenção dos sistemas desenvolvidos com tecnologia orientada para objectos, implicam a diminuição dos custos afectos à construção e aperfeiçoamento dos sistema informáticos.

3.4.2 Eventuais problemas.

A tecnologia orientada para objectos não atingiu ainda um estado estável, verificando-se uma coexistência, nem sempre pacífica, de diferentes perspectivas. Se do ponto de vista académico e científico este facto é interessante e salutar, sendo sintomático de

um processo de permanente evolução, numa perspectiva comercial, o fenómeno reflecte-se em alguma insegurança do mercado na adopção desta tecnologia.

A escassez de ferramentas de desenvolvimento adequadas (como por exemplo, OOCASEs e linguagens de 4ª geração orientadas para objectos) é igualmente, no presente momento, um factor desincentivador.

Para a concepção e desenvolvimento de sistemas informáticos com utilização de tecnologia orientada para objectos é fundamental a contratação ou formação de pessoal qualificado. O processo de adopção na organização de novas técnicas e métodos, de forma obter vantagens inerentes ao usufruto desta tecnologia, requer largos meses ou mesmo anos de utilização activa (Taylor, 1991a, p.112).

A concepção de modelos orientados para objectos apresenta maior complexidade, decorrente da própria complexidade do conceito de objecto e da intensa aplicação do processo de abstracção. A construção de classes de objectos tende a ser um trabalho restrito, realizado por profissionais especializados, limitando-se a generalidade dos programadores à incorporação e interligação dessas classes nos respectivos sistemas.

É igualmente importante o investimento em classes reutilizáveis, assim como é conveniente adoptar uma perspectiva de desenvolvimento através da rápida construção de protótipos (Taylor, 1991a).

O desenvolvimento de "bibliotecas" de código, com classes reutilizáveis que incorporem a natureza inerente ao funcionamento do sistema de informação organizacional, é fundamental. Este facto poderá implicar a realização de um projecto de adaptação do código existente que, apesar de eventualmente suportado por tecnologias obsoletas, não deve ser subestimado, pois representa normalmente anos de aperfeiçoamento e adaptação à estrutura da organização (Bonar, 1990).

Todos os intervenientes directos no processo de informatização, assim como os próprios executivos responsáveis, devem compreender a natureza desta tecnologia, pois ela reflecte-se em todas as fases do ciclo de vida do sistema informático.

A definição de uma política de utilização da tecnologia orientada para objectos implica inicialmente assinaláveis custos de reconversão, pois representa a adopção de um novo paradigma, e requer um envolvimento global de recursos humanos, técnicos e financeiros. A não sensibilização dos intervenientes para este facto poderá conduzir ao tão indesejável fracasso.

Capítulo IV

A Análise Orientada para Objectos

4.1 Caracterização da abordagem e critérios de avaliação.

A tecnologia orientada para objectos fluiu no sentido da concepção de novos métodos de análise e desenho de sistemas. Os métodos de análise orientada para objectos proliferaram nos últimos anos, apesar de por vezes insuficientemente descritos e formalizados. Existem igualmente vários trabalhos publicados com o objectivo de avaliar e comparar os diferentes métodos existentes (Arnold *et al.*, 1991; Cribbs *et al.*, 1992; De Champeaux e Faure, 1992; Fowler, 1992; Goor *et al.*, 1992; Monarchi e Puhr, 1992). Estes trabalhos analisam as características técnicas dos referidos métodos, procurando concluir sobre a sua eficiência na construção de aplicações informáticas.

A nossa abordagem é simultaneamente mais restrita e mais abrangente. É mais restrita, porque se centra na automatização de um determinado tipo de sistemas que são os sistemas de informação de gestão. É mais abrangente, porque vamos procurar analisar o impacto da utilização de métodos de análise orientada para objectos, não apenas na construção de aplicações, como principalmente equacionar o reflexo da sua utilização na eficácia do Sistema de Informação de Gestão. Procuramos desta forma introduzir uma perspectiva organizacional nesta problemática.

Um método de análise de sistemas de informação deve permitir, através da utilização das suas técnicas, a construção de um modelo que represente a realidade organizacional de uma forma tão fidedigna quanto possível, não atendendo a condicionalismos provenientes de detalhes de implementação que possam eventualmente viciar essa representação.

A função da fase de análise de sistemas é documentar o sistema de informação a automatizar ou a conceber, transcrevendo através das suas especificações a realidade inerente ao sistema organizacional, que se irá reflectir no sistema computacional a desenvolver.

A comunicação entre os utilizadores do sistema de informação e os informáticos que vão automatizar esse sistema é fundamental. O modelo de análise é um suporte documental essencial para que essa comunicação se realize sem distorções.

Nem sempre existe, por parte dos diferentes autores, uma distinção clara entre os conceitos de análise e desenho. Por outro lado, alguns dos métodos, na tentativa de acelerar o processo de desenvolvimento de *software*, procuram uma maior integração e continuidade entre as diferentes fases, esbatendo eventuais diferenças de delimitação.

Em nossa opinião, a fase de análise de sistemas de informação não deve, nem pode, ser fundida (ou confundida) com a fase de desenho. A fase de desenho de sistemas, tal como ela é entendida nos actuais métodos de desenvolvimento de *software*, quer eles sejam estruturados ou orientados para objectos, não é, a rigor, uma fase de desenho do sistema de informação, mas sim uma fase de desenho de um sistema computacional adequado à orgânica de funcionamento do sistema de informação. O desenho de sistemas deve ser sempre condicionado aos requisitos definidos durante a fase de análise.

O facto de se pretender acelerar o processo de desenvolvimento, através da fusão das fases de análise e desenho, poderá ter reflexos negativos no desenvolvimento de *software*, principalmente na automatização de sistemas organizacionais de grande dimensão e elevado grau de complexidade. Corre-se o risco de o sistema informático a construir não ser o mais adequado às necessidades da organização. Os requisitos e especificações, resultantes da fase de análise de sistemas de informação, podem ser submergidos pela realização de um desenho orientado no sentido da rápida concepção de um sistema informático.

Neste estudo vamos incluir um método, da autoria de Wirfs-Brock *et al.* (1990), denominado de desenho de sistemas. Apesar deste método não pressupor a existência de uma fase com a designação de "análise de sistemas", apresenta, em termos de conteúdo, muitas semelhanças com outros métodos existentes designados de análise orientada para objectos.

A nossa apreciação dos métodos de análise orientada para objectos vai ter como parâmetros de avaliação os seguintes aspectos:

a) Incorporação no método de uma filosofia de orientação para objectos.

b) Capacidade de representação organizacional das técnicas utilizadas no método.

c) Nível de integração da fase de análise no processo de desenvolvimento de *software*.

d) Capacidade de gestão da complexidade, através de mecanismos de decomposição funcional e interligação de subsistemas organizacionais.

e) Capacidade de o modelo resultante da fase de análise ser entendido pelos utilizadores do sistema de informação e responsáveis pela gestão, de forma a validarem a sua construção e contribuírem para o seu aperfeiçoamento.

f) Capacidade de identificar a informação necessária para os sistemas de informação de gestão, aos níveis estratégico, táctico e operacional.

De entre as dezenas de métodos de análise orientada para objectos existentes, vamos abordar aqueles que julgamos mais significativos e que estão publicados de uma forma relativamente completa (normalmente em livro). A descrição que vamos apresentar destes métodos não pretende ser completa e exaustiva, para isso existem as publicações originais. Pretende sim salientar as suas principais características e diferenças, e, principalmente, analisar a capacidade destes métodos em satisfazer os requisitos expressos nas alíneas anteriores.

Este capítulo irá culminar com uma matriz de avaliação dos métodos abordados, de acordo com o conjunto de requisitos que, em nosso entender, um método de análise de sistemas de informação deve contemplar.

Em anexo, encontra-se um resumo da notação utilizada por cada um dos métodos.

4.2 Métodos e técnicas de Análise Orientada para Objectos

4.2.1 - Análise Orientada para Objectos - Coad e Yourdon.

O método de análise orientada para objectos proposto por Coad e Yourdon (1991a), consiste nos seguintes passos:

a) Encontrar Classes e Classes de Objectos [1].
b) Identificar Estruturas.
c) Identificar Assuntos.
d) Definir Atributos.
e) Definir Serviços.

As técnicas utilizadas são um diagrama principal, designado de "diagrama de classes e objectos", complementado com o diagrama de estado do objecto e o diagrama de serviços. O método proporciona ainda, para cada classe, uma especificação descritiva dos respectivos atributos e serviços e sugere a eventual necessidade de construção de uma tabela de serviços/estados que represente a dependência dos serviços disponíveis em relação aos diferentes estados em que um objecto se pode situar.

Para determinar quais as classes e classes de objectos existentes, sugere-se uma observação global do sistema de informação, aprofundada através do diálogo com especialistas que conheçam o domínio do sistema a automatizar, e a consulta de resultados de anteriores trabalhos de análise efectuados nesse sistema de informação ou em sistemas semelhantes.

As estruturas citadas na alínea b) são basicamente de dois tipos: generalização/especialização e agregação ("todo-parte")[2]. É igualmente assinalada a

---

[1] As "classes de objectos" são designadas no original de "Class & Object", referindo as classes que possuem objectos. O método contempla a possibilidade de existência de classes que não tem directamente objectos. Por exemplo, numa relação de generalização poderão apenas existir objectos para as classes especializadas, tendo a classe genérica apenas como função expressar com maior rigor semântico a realidade inerente ao universo em análise.

[2] A definição adoptada de generalização e agregação encontra-se na pág. 54.

possibilidade de combinar várias estruturas de generalização e agregação, e a representação de situações de múltipla hereditariedade.

No sentido de gerir a extensão ou complexidade do domínio do sistema de informação, expresso através da identificação de um elevado número de classes, propõem-se a partição desse sistema em diversos subsistemas interligados, delimitados a partir de "assuntos" ("subjects"). Esta partição não será necessária em sistemas de informação de reduzida dimensão.

O método sugere que, após a identificação das classes, estruturas e assuntos, se determinem quais os atributos adequados para as classes existentes. Os atributos definem as propriedades relativas aos objectos e permitem relacionar classes entre si através das respectivas instâncias (associação de classes de objectos).

O conjunto de estados pelos quais um objecto pode passar e a dinâmica de transição entre esses estados são graficamente representados através de uma técnica designada de diagrama de estado do objecto. A adopção de um determinado estado, por parte de um objecto, reflecte-se directamente no valor dos seus atributos.

O diagrama de serviços é uma técnica semelhante ao fluxograma, utilizada para especificar os algoritmos relativos aos serviços prestados pelas classes [3]. Entende-se por "serviço" o comportamento exibido por um objecto e que é função dos métodos incorporados na classe à qual esse objecto pertence. Os serviços disponíveis são classificados em algoritmicamente simples e algoritmicamente complexos. Os serviços algoritmicamente simples incluem operações relativamente elementares, como, por exemplo, criar, actualizar ou apagar.

Este método de análise incorpora as características básicas do paradigma da orientação para objectos, como a capsulação de dados e processos, a existência de serviços e o envio de mensagens. As suas representações apresentam uma tendência para a preponderância de estruturas hierárquicas de generalização e de agregação.

Um aspecto que consideramos importante salientar é a não elaboração, pelo menos de uma forma regular, de diagramas de fluxos de dados ou de outros diagramas com funções semelhantes (como os diagramas do modelo funcional do OMT (Rumbaugh *et al.*, 1991) e os diagramas de fluxos de objectos (Martin e Odell, 1992)). Como os autores referem, "Embora reconhecendo a importância e a relevância de muitos outros métodos mais antigos, nós agora deixamo-los para trás, nenhum dos autores desenhou um único diagrama de fluxos de dados nos últimos anos, excepto (em raras ocasiões) como meio de partição de serviços

[3] Ver notação, em anexo, na pág. 115.

complexos num modelo de análise orientada para objectos" (Coad e Yourdon, 1991a, p.193). Apesar de na citação se expor claramente que a elaboração de DFDs é desnecessária, assinala-se igualmente a sua eventual utilidade como mecanismo de gestão da complexidade, nomeadamente de serviços complexos.

O diagrama de serviços corresponde à indispensável perspectiva funcional, mas a técnica utilizada na sua construção, cujas raízes se encontram nos conhecidos fluxogramas, representa em nosso entender um retrocesso tecnológico. A elaboração de fluxogramas foi há muito abandonada pelos principais métodos de desenvolvimento, dando lugar à especificação de processos através de pseudocódigo ou outras linguagens textuais formais.

O facto de se passar da utilização de métodos estruturados para métodos orientados para objectos não nos parece justificativo para a adopção de uma representação gráfica em detrimento de uma representação textual. O código, ou pseudocódigo, a utilizar na especificação dos algoritmos, terá sim de obedecer à concepção global inerente às linguagens de programação orientadas para objectos, como alternativa às especificações estruturadas. A utilização de uma linguagem textual (a exemplo do que acontece com o SOM (Velho, 1991)) permite mais facilmente a geração de código executável através de ferramentas CASE.

Por outro lado, os diagramas de serviços não suportam a decomposição funcional, pelo que se coloca a questão de conhecer como é que um método sem um mecanismo formal de concepção de vários níveis de detalhe poderá suportar a análise de sistemas de informação organizacionais que incluem um elevado número de processos e fluxos de dados.

A relativa simplicidade do método facilita a sua compreensão, mas obviamente limita a sua capacidade de representação. O desenvolvimento da notação do diagrama de *classes e objectos* e o enriquecimento do diagrama de estados afigura-se necessário, paralelamente à substituição do diagrama de serviços por uma outra técnica de especificação de processos.

Segundo os autores, a análise orientada para objectos deverá ter continuidade através das fases de desenho, implementação e teste, orientadas para objectos. A fase de desenho define classes de objectos adicionais, reflectindo a implementação de requisitos, e está dependente da linguagem de programação a utilizar.

4.2.2 OMT (Object Modeling Technique) - Rumbaugh *et al.*

O OMT (Blaha *et al.*, 1988; Rumbaugh *et al.*, 1991) é um método que cobre as fases de análise e desenho do ciclo de vida do desenvolvimento de *software*. A exemplo do que acontece com alguns outros métodos, o OMT centra-se na análise e desenho de um sistema informático.

A fase de análise desenvolve-se a partir de uma descrição do problema a resolver e das características globais do sistema informático pretendido. Deverá emitir um modelo formal que capte os aspectos essenciais relativos a esse sistema, que são a sua estrutura estática (modelo de objectos), sequência de interacções (modelo dinâmico) e transformações de dados (modelo funcional).

O OMT apresenta um conjunto de passos bem definidos para a realização da análise, embora se entenda que a análise de sistemas nem sempre se deve efectuar através de uma sequência rígida de etapas. É por vezes necessário, em sistemas de grande dimensão, um aperfeiçoamento continuo e interactivo dos seus modelos de suporte (Rumbaugh *et al.*, 1991, p.149).

O modelo de objectos pressupõe a realização dos seguintes passos:

a) Identificar objectos e classes.

b) Elaborar um dicionário de dados.

c) Determinar quais as relações existentes entre os objectos.

d) Identificar atributos.

e) Estruturar hierarquicamente as classes.

f) Verificar se o modelo permite responder às questões mais relevantes, e em caso negativo, completá-lo com a informação necessária para que possa responder.

g) Agrupar as classes em módulos de dimensão aproximadamente uniforme.

O modelo dinâmico expressa a dependência do comportamento do sistema em relação ao tempo. É necessário identificar quais os eventos que afectam o sistema, e representar sequencialmente a sua ocorrência, para cada classe de objectos, num diagrama de estados.

A elaboração do modelo dinâmico contempla as seguintes etapas:

a) Preparar cenários correspondentes às interacções esperadas com o sistema.

Cada cenário corresponde a uma sequência de eventos. Um evento ocorre sempre que existir troca de informação entre um objecto do sistema e um agente externo. Ao cenário inicial, que consiste nas acções consideradas usuais, vai-se acrescentar as condições necessárias inerentes ao funcionamento desse sistema, assim como as omissões e erros que possam ser cometidos pelos utilizadores.

b) Definir o formato dos *interfaces* com o utilizador.

c) Identificar quais os eventos relacionados com cada cenário.

d) Construir um diagrama de estados para cada classe de objectos, desde que o seu comportamento dinâmico não seja trivial. O diagrama de estados traduz-se graficamente numa rede de diferentes estados, modificados através da ocorrência de eventos.

e) Verificar a consistência do modelo.

O modelo funcional permite representar como se realizam os processos de computação e exprimir a dependência funcional existente.

Para a construção do modelo funcional é necessário:

a) Identificar os dados que entram e saem do sistema.

b) Construir um diagrama de fluxos de dados que represente como é que os dados de entrada são armazenados e processados, dando origem aos *outputs* desse sistema.

c) Especificar cada função recorrendo a uma linguagem natural, equações matemáticas ou pseudocódigo.

d) Identificar constrangimentos entre os objectos.

O funcionamento do sistema deve obedecer a um determinado conjunto de condições que podem estar simultaneamente relacionadas com mais do que um objecto e que devem ser incorporadas nos modelos dinâmico e funcional.

e) Especificar critérios de optimização.

Um critério possível poderá ser, por exemplo, minimizar o número de mensagens enviadas entre diferentes locais, ou minimizar o tempo de resposta do sistema informático (Rumbaugh *et al.*, 1991).

O modelo de objectos, ao contrário do que tradicionalmente acontece com os modelos de dados nos métodos de análise estruturada, desempenha normalmente um papel principal e expressa de forma sumária operações com origem nos outros dois modelos. A importância atribuída a cada um dos modelos pode variar de acordo com o tipo de sistema

que se pretende construir (Rumbaugh *et al.*, 1991, p.149). Por exemplo, em problemas que envolvam interacção e tempo de resposta, o modelo dinâmico assume assinalável relevância.

O OMT não é um método puramente orientado para objectos. O facto de apresentar um modelo funcional independente do modelo de objectos é sintomático de alguma desagregação entre a representação de dados e processos, e revela fundamentos nos métodos estruturados. No modelo de objectos, os atributos representam simples valores e não são entendidos como objectos (Rumbaugh *et al.*, 1991, p.23). Em algumas abordagens orientadas para objectos, o valor de um atributo pode ser considerado como um objecto de uma classe. Por exemplo, o valor numérico 5 pode ser entendido como um objecto da classe dos números inteiros.

Desenvolvido com o intuito de possibilitar uma maior capacidade de expressão no desenho de bases de dados relacionais (Blaha *et al.*, 1988), o OMT apresenta algumas características com origem no modelo relacional, como por exemplo, a existência de "classes associativas" que incluem atributos e métodos de ligação entre duas ou mais classes. Segundo os autores, a utilização de técnicas de análise e desenho orientadas para objectos transcende a escolha do tipo de SGBD (hierárquico, reticulado, relacional ou orientado para objectos) a utilizar (Rumbaugh *et al.*, 1991, p.366).

O modelo de objectos define com assinalável detalhe a forma de modelar situações relativamente específicas, possuindo para tal os mecanismos usuais de relação entre classes (associação, generalização e agregação), com alguns "refinamentos" peculiares como as associações qualificadas.

Uma associação qualificada relaciona dois objectos através de um qualificador. O qualificador é um atributo especial utilizado para reduzir a multiplicidade efectiva de uma associação (Rumbaugh *et al.*, 1991, p.35).

A notação utilizada no modelo de objectos permite estabelecer restrições para os objectos, classes, atributos e relações, assim como representar a "propagação de operações". A "propagação" corresponde à aplicação automática de uma operação a uma rede de objectos de diferentes classes quando essa operação é inicialmente activada para um objecto dessa rede (Rumbaugh *et al.*, 1991, p.60).

O modelo dinâmico introduz os conceitos de evento, estado, actividade e acção.

Um evento corresponde a um estímulo externo aplicado sobre um objecto, com possível repercussão interna, implicando a ocorrência de um outros estímulos.

Os estados definidos para um objecto determinam o valor dos atributos e as operações desse objecto. Os eventos, ao afectarem o estado de um objecto, alteram o valor dos atributos e implicam a realização de actividades. Uma actividade é uma operação contínua, inserida num estado, que é executada quando o objecto assume esse estado.

Uma acção é definida como uma operação de duração insignificante que está associada à ocorrência de um evento. Por exemplo, se o utilizador premir a tecla *x* (evento) a aplicação deverá imediatamente colocar o menu *y* no ecran (acção). As acções podem representar operações de controlo interno, como, por exemplo, atribuir valores a atributos ou gerar a ocorrência de outros eventos (Rumbaugh *et al.*, 1991, p.93).

As estruturas hierárquicas de generalização e agregação aplicam-se não apenas ao modelo de objectos como também ao modelo dinâmico.

A generalização de estados revela-se de considerável interesse e apresenta uma correspondência directa com a hierarquia de classes do modelo de objectos.

No que diz respeito aos eventos, estes não possuem, por natureza, um elevado grau de complexidade interna, pelo que a sua estruturação através do recurso a mecanismos de generalização não se revela muito significativa.

O comportamento de um sistema pode ser definido pelo agregado de diagramas de estados dos seus componentes, neste caso, das diferentes classes de objectos que o constituem.

O modelo funcional do OMT permite especificar o sentido das operações incluídas no modelo de objectos e das acções do modelo dinâmico. Este modelo baseia-se na elaboração de diagramas de fluxos de dados, sendo a notação utilizada relativamente semelhante à inicialmente definida por De Marco (1978).

Apesar da semelhança de notação, os DFDs em Rumbaugh *et al.* não são apenas utilizados numa perspectiva de representação do sistema de informação da organização, estendendo a sua contribuição para a esquematização funcional do sistema computacional a desenvolver. O DFD "mostra a relação funcional entre os valores computados pelo sistema" (Rumbaugh *et al.*, 1991, p.124).

O conceito de entidade externa, que caracteriza os tradicionais diagramas de fluxos de dados, é substituído pelo conceito de "actor". Um actor consiste num " objecto activo que conduz o diagrama de fluxos de dados produzindo ou consumindo valores" (Rumbaugh *et al.*, 1991, p.126). Como exemplos de actores, podemos ter: um "cliente", um "cartão de crédito" ou um "*buffer* de ecran" (Rumbaugh *et al.*, 1991).

O OMT socorre-se dos diagramas de fluxos de dados para representar detalhes de implementação. É frequente a sua utilização numa perspectiva informática, relegando para segundo plano a representação organizacional.

Apesar de se assinalar a possibilidade de efectuar a decomposição funcional através de diferentes níveis de detalhe, o OMT não apresenta uma notação que clarifique a forma de ligação desses níveis.

A relação entre as fases de análise e desenho é natural e fluente. O desenho realiza-se a partir dos modelos obtidos na análise e consiste em:

" - Combinar os três modelos para obter operações nas classes.
- Desenhar algoritmos para implementar operações.
- Optimizar os caminhos de acesso aos dados.
- Implementar o controlo para interacções externas.
- Ajustar a estrutura das classes de forma a reforçar a hereditariedade.
- Desenhar associações.
- Determinar a representação de objectos.
- Agrupar as classes e associações em módulos. "

(Rumbaugh *et al.*, 1991, p.228)

As fases de análise e desenho apresentam uma forte ligação entre si, porque ambas tem como objecto um sistema computacional.

O OMT apresenta uma notação clara e concisa. É um método equilibrado, analisando paralelamente as perspectivas dinâmica, funcional e estrutural, dotado de técnicas desenvolvidas e interligadas que possibilitam uma especificação detalhada e consistente do sistema computacional a desenvolver, através de uma abordagem orientada para objectos, denotando algumas características com origem na análise estruturada de sistemas.

4.2.3 Desenho baseado em responsabilidades - Wirfs-Brock *et al.*

O trabalho de Wirfs-Brock *et al.* (1990) não contempla a existência de uma fase designada de análise, passando da especificação de requisitos para o desenho. Este método assume claramente a ênfase na concepção do sistema informático. A sua fase de desenho incorpora aspectos que outros métodos orientados para objectos atribuem à análise de sistemas de informação.

A abordagem de Wirfs-Brock apresenta características peculiares. Centra-se na definição de responsabilidades atribuídas às classes de objectos e na identificação das necessidades de colaboração entre essas classes.

As responsabilidades de um objecto "são todos os serviços que ele proporciona para os contratos que suporta" (Wirfs-Brock *et al.*, 1990, p.62). Um contrato corresponde à definição de uma lista de serviços que um objecto de uma classe pode requerer a um objecto de outra classe. O objecto que requer o serviço designa-se de *cliente* e o objecto que recebe a mensagem, e que coloca à disponibilidade do cliente o serviço, designa-se de *servidor* (*server*) (Wirfs-Brock *et al.*, 1990, p.31). Um serviço é entendido como a realização de uma acção ou a transmissão de determinada informação.

O facto de se identificar uma classe indica que existe uma necessidade que pode ser preenchida por essa classe, e portanto, ela terá pelo menos uma responsabilidade. A tentativa de cumprir as responsabilidades tem como possível consequência a colaboração entre as classes.

Nem todas as responsabilidades são passíveis de inclusão em contractos. Algumas responsabilidades, que não podem ser requisitadas por outros objectos, designam-se de responsabilidades privadas (Wirfs-Brock, 1990, p.117).

A identificação de classes é realizada tendo como suporte a utilização de cartões CRC ("Class, Colaboration and Responsability")[4]. Neste cartão é colocado o nome da classe, a sua categoria (abstracta ou concreta), as suas responsabilidades, as classes que com ela colaboram, e uma lista das respectivas subclasses e superclasses. Esta técnica foi inicialmente desenvolvida com objectivos didácticos por Beck e Cunningham (1989).

A hierarquia de classes é representada recorrendo a gráficos hierárquicos, diagramas de Venn e contratos.

---

[4] Ver notação gráfica, em anexo, na pág. 125.

Num gráfico hierárquico, as classes são simbolizadas através de rectângulos com a designação da classe. As superclasses (o método contempla a possibilidade de definir estruturas de múltipla hereditariedade) são desenhadas acima das subclasses e ligadas através de segmentos de recta. O canto superior esquerdo do rectângulo referente à classe é preenchido no caso de se tratar de uma classe abstracta.

Os diagramas de Venn representam as responsabilidades de cada classe através de uma elipse. A intercepção de elipses indica que existem responsabilidades comuns a diferentes classes, as quais deverão ser expressas através da definição de uma superclasse [5].

Um contrato representa um conjunto coeso de responsabilidades. Ao agruparem-se as responsabilidades de uma classe que são utilizadas pelos mesmos clientes, poderá deduzir-se que essas responsabilidades são sempre utilizadas em conjunto, implicando a concepção de uma estrutura de generalização.

Os subsistemas são identificados encontrando um conjunto de classes que individualmente possuem diferentes responsabilidades, mas cuja colaboração conjunta permite preencher uma responsabilidade de nível mais elevado.

A utilização de cartões também se aplica aos subsistemas. Cada cartão de subsistema deverá indicar a designação desse subsistema, uma descrição da sua função, uma lista dos contratos requeridos por clientes externos e a identificação das classes internas que suportam a realização dos respectivos contratos (Wirfs-Brock *et al.*, 1990, p.137).

Os "diagramas de colaboração" expressam como as classes de objectos (ou os subsistemas) colaboram entre si para responderem às suas responsabilidades.

A comunicação através de mensagens entre objectos obedece a um protocolo de comunicação. Um protocolo corresponde a " um conjunto de sinais aos quais a classe irá responder" (Wirfs-Brock *et al.*, 1990, p.162). Esses sinais incluem o nome do método a operar, os respectivos parâmetros, e o tipo de objecto que resulta da execução do método.

O método de Wirfs-Brock *et al.* propõe os seguintes passos:

a) Identificação de classes.

b) Identificação de responsabilidades.

c) Identificação da colaboração entre as classes.

d) Construção de gráficos de hierarquia.

e) Delimitação de subsistemas.

---

[5] Ver notação gráfica, em anexo, na pág. 126

f) Construção de protocolos para cada classe.

Segundo Fowler (1992, p.21), a técnica inicial de identificação de classes, baseada na noção de responsabilidade, é em geral bem aceite e largamente recomendada. A notação utilizada é relativamente intuitiva e possibilita, através do uso de cartões CRC, uma introdução natural a uma representação orientada para objectos.

O método não contempla a elaboração de diagramas de fluxos de dados nem de diagramas de transição de estados, o que contribui para uma insuficiente representação dinâmica do sistema de informação a automatizar. O comportamento do sistema é formulado em termos de contratos, responsabilidades e mensagens.

A técnica de identificação e interligação de subsistemas é relativamente simples e normalmente aplicável ao desenho de módulos de *software*. Contudo, é possível a sua aplicação na partição de subsistemas organizacionais.

O trabalho de Wirfs-Brock *et al.* (1990) apresenta uma perspectiva interessante na utilização dos princípios da orientação para objectos. As técnicas deste método possuem um elevado potencial, mas estão fundamentalmente dirigidas para a concepção de um sistema informático e não para as especificações do sistema de informação, o que é perfeitamente natural num método de desenho de *software*.

4.2.4 Análise Orientada para Objectos - Shlaer e Mellor

Shlaer e Mellor (1988; 1992) apresentam um método designado de análise orientada para objectos que surgiu do esforço desenvolvido pelos autores no aperfeiçoamento de técnicas de análise estruturada.

Este método tem como suporte a elaboração de três modelos: modelo da informação (*information model*), modelo de estados e modelo de processos; desenvolvidos através das seguintes etapas:

a) Construção do modelo da informação.

b) Construção de um modelo de estados.

c) Construção de um modelo de processos.

d) Integração dos três modelos.

O modelo da informação é fundamentalmente composto por diagramas de estrutura da informação, complementados com a descrição dos atributos dos objectos e uma lista das relações existentes entre esses objectos.

A modelação de dados apresentada na primeira publicação (1988) é relativamente elementar. O diagrama de estrutura da informação é uma técnica baseada no modelo entidade-associação de Chen (1976), assinala a existência de estruturas de generalização e relações binárias ou ternárias entre os objectos, mas não aprofunda as estruturas de agregação e a múltipla hereditariedade.

Na sua mais recente publicação (1992), Shlaer e Mellor desenvolvem significativamente o modelo da informação, acrescentando alguns conceitos interessantes, como a composição de relações entre classes de objectos.

Apesar do seu recente aperfeiçoamento, a orientação para objectos deste método é questionável. O conceito de objecto definido em Shlaer e Mellor (1988) é discutível, assemelhando-se ao conceito geralmente aceite de entidade.

O modelo de estados, que referencia a evolução temporal dos objectos e assinala os fluxos de controlo da informação, está bem concebido e apresentado. As técnicas utilizadas são o diagrama de transição de estados, as tabelas de transição de estados, a descrição das acções do diagrama de transição de estados, a descrição de eventos e o modelo de comunicação entre objectos (Shlaer e Mellor, 1992).

O diagrama de transição de estados deve ser elaborado para os objectos e para as relações expressas no modelo da informação. A sua notação é composta por rectângulos representando os estados, interligados através de setas que simbolizam as transições de estado em virtude da ocorrência de eventos.

A elaboração do modelo de comunicação entre objectos surge da necessidade de expressar a coordenação na ocorrência de eventos que afectam esses objectos.

O modelo de processos fundamenta-se na elaboração de diagramas de fluxos de dados e tem como objectivo de detalhar as acções referidas no modelo de estados.

A integração dos três modelos é realizada a partir do modelo da informação, tendo em atenção os seguintes aspectos:

- Os diagramas de transição de estados são construídos para os objectos que exibem um ciclo de vida, ou um ciclo operacional.

- As transições de estados implicam a realização de acções. Essas acções são representadas como processos no diagrama de fluxos de dados.

- Cada objecto do modelo da informação transforma-se num arquivo no diagrama de fluxos de dados.

- Os processos aceitam e produzem apenas os dados definidos no modelo da informação.

- Os processos podem criar eventos que se representam no diagrama de transição de estados.

(Shlaer e Mellor, 1988, p.97)

Apesar da primeira publicação de Shlaer e Mellor (1988) que aborda o método de análise em questão se designar de "Análise Orientada para Objectos", pensamos que a noção de orientação para objectos não foi correctamente entendida pelos autores. O referido trabalho não inclui os mais elementares princípios de orientação para objectos, como, por exemplo, a capsulação de dados e processos.

Não concordamos com Rumbaugh *et al.* (1991, p.273) quando afirmam que Shlaer e Mellor "descrevem um método completo para análise orientada para objectos que é semelhante ao nosso". Em nossa opinião, o referido método é fundamentalmente um método de análise estruturada sendo incorrecta a sua classificação como orientado para objectos. Apesar de ambos os métodos possuírem técnicas com objectivos semelhantes, subdividindo ambos o processo de análise em três fases (modelação de dados, modelação dinâmica de

estados e eventos, e modelação funcional), enquanto que o OMT incorpora os princípios fundamentais de orientação para objectos, o método de Shlaer e Mellor não o faz.

A evolução evidenciada em Shlaer e Mellor, entre uma abordagem estruturada para uma abordagem orientada para objectos, é mais sintáctica do que semântica. Como afirmam Winsberg e Richards (1991, p. 33), a eventual substituição no modelo da informação da palavra "objecto" por "entidade" iria impossibilitar uma distinção conceptual deste modelo com qualquer abordagem baseada no diagrama entidade-associação.

4.2.5 Análise Orientada para Objectos - Martin e Odell.

Segundo Cribbs *et al.* (1992) e Fowler (1992), o método de análise orientada para objectos apresentado por Martin e Odell (1992), que vamos abordar, foi concebido com base no método Ptech da autoria de John Edwards e desenvolvido na Associative Design Technology.

Este método, proposto por Martin e Odell, subdivide a fase de análise em duas grandes áreas: a análise estrutural e a análise comportamental dos objectos.

A análise estrutural baseia-se no diagrama de relação de objectos e tem como objectivo definir as diferentes classes de objectos perceptíveis no sistema real, assim como determinar as relações a estabelecer entre essas classes, através das estruturas de generalização, agregação ("composição", segundo Martin e Odell) e associação. O diagrama de relação de objectos ("object-relationship") é basicamente semelhante ao modelo entidade-associação ampliado[6] (Martin e Odell, 1992, p.83). A notação utilizada é expressiva e detalhada. Contempla a possibilidade de representar múltiplos conjuntos de diagramas relacionados, subdividindo o esquema inicial caso este se apresente muito complexo e de difícil compreensão.

Por sua vez, a análise comportamental centra-se na evolução dos objectos ao longo do tempo. Esta perspectiva é obtida através de esquemas de eventos, diagramas de dependência de processos e diagramas de transição de estados.

Os esquemas de eventos são apropriados para descrever os processos através de eventos, *triggers*[7] , condições e operações. Um evento pode ser classificado como interno, temporal (*clock event*) ou externo.

Os eventos internos são resultado de operações executadas pelos objectos do sistema. Os eventos temporais são produzidos periodicamente ou após decorrer um determinado período de tempo. Os eventos externos são aqueles que têm origem fora das fronteiras do sistema mas que são susceptíveis de influenciar esse mesmo sistema.

---

[6] Entendemos por modelo entidade-associação ampliado, o modelo entidade associação proposto por Chen (1976), com inclusão das estruturas de generalização e agregação (definidas originalmente por Smith e Smith (1977)).

[7] Um *trigger* corresponde a um dispositivo que funciona através de uma relação de causa-efeito, em que a execução imediata e automática de uma operação está condicionada à ocorrência de um determinado evento e à existência de condições previamente definidas.

Os diagramas de transição de estados (designados de *fence diagrams*) possibilitam uma visão simplificada e concisa dos possíveis estados existentes e de como esses estados se relacionam.

Para representar sistemas de informação extensos e complexos é assinalado o facto de ser necessário dispor de uma técnica que possibilite um alto nível de compreensão do sistema de informação na sua totalidade. Neste caso, é recomendada a construção de diagramas de fluxos de objectos (Martin e Odell, 1992, p.101).

Os diagramas de fluxos de objectos têm como objectivo delinear a um elevado nível de abstracção a área do problema em análise. Derivam da construção dos tradicionais diagramas de fluxos de dados, incluindo a noção de decomposição funcional, e devem modelar o espaço do problema e não a solução computacional desse mesmo problema.

Esta técnica foi desenvolvida no sentido de representar a orgânica de funcionamento do sistema de informação com alguma analogia a um sistema de transformação industrial. As actividades ou processos são entendidas como produtoras e consumidoras de produtos. O produto é o resultado final que justifica a existência e o propósito de uma actividade. Os produtos são produzidos para posteriormente serem consumidos por outras actividades, gerando estas novos produtos com valor acrescentado.

A identificação de cadeias de valor é uma técnica já anteriormente aplicável no planeamento estratégico (Porter, 1985) e agora utilizada na análise de sistemas para representar funcionalmente o sistema de informação a um nível operacional.

A distinção entre os diagramas de fluxos de dados e os diagramas de fluxos de objectos reside fundamentalmente no facto de, nestes últimos, os objectos produzidos e consumidos não serem apenas dados. Os diagramas de fluxos de objectos devem representar "qualquer tipo de coisa que passe de uma actividade para outra: ordens, componentes, bens finais, gráficos, serviços, *hardware*, *software* - ou dados" (Martin e Odell, 1992, p.101). Esta perspectiva acompanha uma tendência evolutiva evidenciada em outros métodos de análise de sistemas de informação, como por exemplo o SSADM que, na sua versão 4, contempla a possibilidade de representar a circulação de bens físicos através de DFDs (Downs *et al.*, 1992).

O método de análise apresentado por Martin e Odell (1992) é um método moderadamente orientado para objectos, apresentando algumas características provenientes na análise estruturada. Não possui, por exemplo, uma técnica de representação simultânea de dados e processos.

Martin e Odell procuram adaptar o método *Information Engineering*, da autoria de James Martin, ao paradigma da orientação para objectos, definindo um modelo de "engenharia da informação orientada para objectos" que inclui as tradicionais fases de

planeamento estratégico da informação, análise, desenho e construção. A interligação entre as fases de análise e desenho está bem definida e articulada, não estando perfeitamente explícito a forma de integração entre a análise de sistemas de informação orientada para objectos e o planeamento estratégico da informação.

4.2.6 Semantic Object Model - Velho.

O *Semantic Object Model* (SOM) incorpora uma estrutura de conceitos, definidos com base no paradigma da orientação para objectos, que são utilizados, através de linguagens de especificação de sistemas, no sentido de construir modelos orientados para objectos representativos do mundo real.

Segundo o autor, " basicamente o SOM explora as sinergias derivadas da combinação entre a perspectiva estrutural dos modelos semânticos e a perspectiva de tipos abstractos de dados adoptada pela orientação para objectos" (Velho, 1991, p.5), procurando, através desta complementaridade, representar os aspectos estruturais e comportamentais inerentes aos sistemas de informação.

Dispõe de duas linguagens de especificação de sistemas: TSOM (linguagem textual) e GSOM (linguagem gráfica). Estas duas linguagens têm funções complementares. Enquanto que o GSOM possibilita a modelação a um nível de abstracção mais elevado, os detalhes de implementação são fundamentalmente explicitados através da linguagem textual, pois esta permite mais facilmente derivar para a implementação do sistema computacional.

O SOM tem como objectivo cobrir através de uma única estrutura de conceitos as fases de análise, desenho e programação do ciclo de desenvolvimento de sistemas de informação automatizados.

Os conceitos de base inerentes a uma abordagem orientada para objectos estão fortemente implantados e detalhadamente definidos e formalizados em SOM.

Um objecto é estruturalmente constituído por um conjunto de componentes, como se pode observar na seguinte transcrição:

> " Em SOM um objecto possui *atributos, eventos, ciclo de vida e invariantes*. Os atributos definem os relacionamentos estruturais com outros objectos. Os eventos ocorrem durante o tempo de vida de um objecto e alteram o valor dos seus atributos. O ciclo de vida define as sequências de eventos permissíveis durante o tempo de vida dum objecto. Os invariantes definem predicados sobre o valor dos atributos os quais têm invariavelmente que se verificar durante todo o tempo de vida de um objecto. " (Velho, 1991, p.25).

Todas as relações estruturais entre as entidades reais são expressas no modelo através de atributos. O valor de um atributo pode ser um objecto de uma classe básica (do tipo: *boolean*, inteiro, carácter, real ou monetário), uma referência ligada a um objecto, um conjunto de referências ligadas a vários objectos ou uma referência vazia (*void*).

Os atributos que estabelecem uma relação estrutural[8] entre objectos podem ser classificados em *designadores, mandatórios, opcionais, associações, inversos, designadores duplos, agregados, componentes, relações, dependentes, possuidores, equivalorados, derivados e pré-definidos,* consoante o tipo de relação que estabelecem e respectivo grau.

O SOM utiliza um só modelo para representar a estrutura e comportamento das entidades do mundo real. Os distintos papeis que uma entidade desempenha ao longo do seu tempo de vida, são representados através de *fases* em que os objectos de uma classe podem entrar e sair, assumindo ou dispensando, nesse momento, o tipo definido nessa fase (Velho, 1991).

Apesar da definição do conceito de fase não ser original, a sua utilização no SOM reveste-se de aspectos peculiares, como sejam a representação conjunta num único modelo de fases e classes de objectos.

A capacidade de expressão semântica desta técnica de modelação é indiscutível. Contudo, entendemos que a sua complexidade torna difícil a compreensão do respectivo modelo por não especialistas. Sendo assim, apresenta algumas limitações para funcionar como suporte documental de comunicação com os gestores e outros utilizadores do sistema de informação. Tal situação é agravada com o facto de não existir (na versão actual do SOM) um mecanismo que possibilite a decomposição do sistema a representar em diversos subsistemas, permitindo a sua percepção através de diferentes níveis de detalhe.

O SOM permite uma interligação praticamente perfeita entre as fases de análise, desenho e implementação. A estrita relação entre as fases de análise e desenho reflecte-se numa sobreposição dos aspectos relativos ao desenho do sistema informático, em prejuízo da representação organizacional que deve ser realizada na fase de análise de sistemas de informação.

O suposto modelo de representação do real, produzido em SOM, é um modelo que é desenvolvido no sentido de ser automatizado. A representação gráfica de um sistema de informação em GSOM inclui um conjunto de evidentes detalhes de implementação, como as designadas classes básicas, o que implica que esta linguagem seja por nós entendida, fundamentalmente, como uma linguagem de desenho de sistemas que, num sentido rigoroso, não permite especificar a finalidade e dinâmica de circulação da informação necessária ao funcionamento de um sistema organizacional.

---

[8] O termo relação é usado em sentido lato, não no sentido definido em SOM em que uma relação é "um atributo que define uma aplicação de muitos-para-muitos a qual pode ser de qualquer grau ..." (Velho, 1991, p.50).

O TSOM apresenta todas as características de uma linguagem de programação orientada para objectos, existindo uma perfeita relação com o GSOM porque ambos representam o mesmo sistema, isto é, ambos documentam um sistema computacional a desenvolver segundo uma perspectiva orientada para objectos.

Se por um lado, a especificação directa em TSOM das operações a executar permite acelerar o processo de desenvolvimento, por outro lado, o grau de dificuldade da sua realização aumenta consideravelmente. Não nos parece recomendável que o levantamento de requisitos e especificações do sistema de informação seja directamente efectuado numa linguagem semelhante a uma linguagem de programação de computador, obviamente mais difícil para o analista do que uma especificação em inglês (ou português) estruturado.

Apesar de ser uma excelente técnica de desenho e programação orientada para objectos, para a sua aplicação ao estudo e concepção de sistemas de informação de gestão, o SOM necessita, em nosso entender, de se interligar a montante com uma técnica de análise que permita a comunicação com os utilizadores e responsáveis organizacionais, e que possibilite documentar as necessidades de informação e fluxos de orientação dessa informação na organização. Esta ligação é necessária para que o sistema informático a desenvolver esteja perfeitamente adequado aos requisitos organizacionais, assegurando a eficiência e eficácia do sistema de informação.

O SOM, na sua versão actual, poderá eventualmente ser directamente aplicável a subsistemas organizacionais quando o carácter operacional e a simplicidade do subsistema não justifique um acentuado dispêndio de tempo na realização da fase de análise, encurtando-se desta forma a duração do ciclo de desenvolvimento. Nesta situação, pensamos que o SOM é eficaz, pois a sua estrutura de conceitos e método de representação estão bem concebidos.

4.2.7 OBject LOGic - Sernadas *et al.*

O OBLOG é uma *abordagem* ao desenvolvimento de sistemas de informação, desenvolvida no INESC a partir de 1986. Segundo os autores, qualquer *abordagem* deverá incluir: " um conjunto de abstracções úteis para a percepção da realidade e para a concepção e implementação dos sistemas de bases de dados; uma estratégia de desenvolvimento de sistemas de base de dados; um conjunto de princípios metodológicos para utilização das abstracções segundo a estratégia em causa" (Sernadas *et al.*, 1989a, p.1.16).

Por *percepção* entende-se a descrição dos aspectos dinâmicos e estáticos dos objectos existentes no mundo real. A *concepção* corresponde à delimitação e especificação dos sistemas computacionais a desenvolver de modo a suportar as actividades de uma empresa (Sernadas, 1989a, p.1.1).

Tal como acontece com o SOM, o OBLOG procura suportar o desenvolvimento de *software* cobrindo de forma integrada as principais fases do ciclo de desenvolvimento - usualmente designadas de análise, desenho e programação. A intensão é gerar semi-automaticamente código de implementação a partir das especificações da sua linguagem, código esse que nunca é manipulado directamente (Sernadas *et al.*, 1992, p.2).

Segundo os autores, o OBLOG é adequado ao desenvolvimento de grandes aplicações, através da reutilização de código, com possibilidade de serem implementadas em arquitecturas abertas e de multi-processador (Sernadas *et al.*, 1992, p.10,11).

O OBLOG possuiu uma notação aprofundada e um conjunto de conceitos devidamente formalizados, onde se incluem os princípios fundamentais do paradigma da orientação para objectos.

A sua linguagem de especificação, que se apresenta rigorosamente definida e com grande poder de expressão, possui uma versão textual e uma versão gráfica ("diagramática", segundo o OBLOG).

A linguagem gráfica baseia-se na elaboração dos seguintes diagramas:

- Diagrama do universo de classes - onde se indicam as classes e respectivas relações entre classes;

- Diagrama matricial de classe - permite declarar os atributos e eventos de cada classe;

- Diagrama de comportamentos de classe - é elaborado um destes diagrama para cada classe, definindo o comportamento permitido a cada instância da classe;

- Diagrama de efeitos de evento sobre atributos - estabelece, para cada tipo de evento de uma classe, quais os efeitos da ocorrência desse evento nos valores dos atributos;

- Diagrama de interacção entre classes - é elaborado um diagrama para cada conjunto de classes cujas instâncias interactuam, indicando que eventos de uma classe correspondem a eventos de outra classe;

- Diagrama de atribuição de valores - é opcional, e serve para estabelecer condições para o valor dos atributos e para os parâmetros dos eventos.

(Sernadas *et al.*, 1992, p.12)

Tal como sucede com a generalidade das abordagens que se concentram fundamentalmente nas estruturas de dados, o OBLOG não dispõe de um bom mecanismo de gestão da complexidade, que permita a visualização dos sistemas de informação a automatizar através de diferentes níveis de detalhe.

O conceito de sistema de informação definido em OBLOG é entendido como sinónimo de sistema informático. A fase de concepção é assinalada como tendo por objectivo a "delimitação e especificação dos sistemas de informação a desenvolver" (Sernadas *et al.*, 1989a, p. 1.1), ou como, a fase em que se "pretende conceber um sistema computacional (ou vários?) para suporte à organização..." (Sernadas *et al.*, 1989a, p. 3.1).

Apesar de, na elaboração do OBLOG, os autores evidenciarem preocupação na utilização das suas técnicas por indivíduos sem formação informática (Sernadas *et al.*, 1992, p.10), o que contribuiu para a introdução de uma representação gráfica, pensamos que o OBLOG não é facilmente acessível a não especialistas. Como um dos autores sublinha, "o período de formação dos analistas é de pelo menos seis meses" (Sernadas, 1993, p.7).

À semelhança do que acontece com outros métodos de análise orientada para objectos, a perspectiva técnica inerente ao desenvolvimento de um sistema computacional sobrepõe-se ao reconhecimento funcional do sistema de informação da organização. O modelo da realidade obtido a partir do OBLOG está, desde a fase de percepção, fundamentalmente orientado para a implementação em computador, não incluindo uma análise da eficiência dos processos organizacionais.

O OBLOG é, contudo, um método bem formalizado e de comprovado valor para concepção e desenvolvimento de sistemas informáticos, possuindo uma potente linguagem de especificação orientada para objectos.

4.2.8 Objectory - Jacobson *et al.*

O Objectory é um método completo de desenvolvimento de sistemas orientado para objectos, cobrindo as fases de identificação e especificação de requisitos, análise, desenho, implementação e teste. Segundo Dave Thomas, este método tem cerca de vinte anos de utilização no desenvolvimento de *software*[9], apesar de serem escassas as publicações que o referem e, normalmente, de forma limitada.

Segundo Jacobson *et al.* (1992, p.193), "o processo de análise ajuda a definir e especificar o sistema a construir" e baseia-se no desenvolvimento de dois modelos: o modelo de requisitos e o modelo de análise.

O modelo de requisitos, que alimenta o modelo de análise, centra-se nos conceitos de *actor* ("actor") e *acontecimento de utilização* ("use case").

Um *actor* corresponde a uma classe de utilizadores[10] do sistema informático, enquanto que um *acontecimento de utilização* consiste na sequência de transacções associadas ao diálogo dos utilizadores com esse sistema. Citando Jacobson *et al.* (1992, p.129), "cada *acontecimento de utilização (use case)* corresponde a uma forma específica de utilizar o sistema e cada execução desse *acontecimento de utilização* pode ser considerada como uma instância do *acontecimento de utilização*".

O Objectory é um método *orientado por acontecimentos de utilização ("*use case driven approach*")*. Quando se pretende alterar o comportamento do sistema informático é necessário reajustar os *actores* e *acontecimentos de utilização* desse sistema.

Definem-se três dimensões para o modelo de análise: comportamento, informação e apresentação (Jacobson *et al.*, 1992, p.134). A apresentação tem como finalidade a compreensão do modelo por não especialistas.

No Objectory, os objectos podem ser classificados em três tipos: objectos entidade, objectos de *interface* e objectos de controlo.

Um objecto entidade incorpora dados e comportamentos e modela a informação relevante para o funcionamento do sistema informático, a qual se mantém a longo prazo. Os objectos entidade correspondem aos elementos usualmente classificados de instâncias de entidade no modelo entidade-associação.

---

[9] Esta afirmação encontra-se no prefácio da autoria de Dave Thomas da obra de Jacobson *et al.*, (1992), *Object-Oriented Software Engineering, A Use Case Driven Approach*, Reading, Addison-Wesley.

[10] Entende-se por utilizador, no contexto do Objectory, todos os indivíduos que utilizam o sistema informático para o desempenho das suas funções.

Um objecto de *interface* modela o comportamento e a informação relativa ao *interface* do sistema computacional. A "ligação" que o utilizador usa para intercomunicar com o sistema informático é, portanto, conceptualmente representada através de objectos de *interface*. De acordo com Jacobson *et al.* (1992, p.170), a função desta categoria de objectos é "traduzir as acções realizadas pelos actores no seu contacto com o sistema, em eventos desse sistema".

Um objecto de controlo modela funcionalidades ou comportamentos que não estão naturalmente associados a outros objectos. Para Jacobson *et al.* (1992, p.135), o seu "comportamento consiste em operar em diferentes objectos entidade, realizando alguma computação, transmitindo então o resultado a um objecto de *interface*".

A razão de ser dos objectos de controlo nasce da constatação que seria forçado, e pouco natural, associar aos outros dois tipos comportamentos que não pertencem ao *interface* e que também não se incluem na informação que é consultada e actualizada durante o funcionamento do sistema computacional e, portanto, incorporada em objectos entidade (Jacobson *et al.*, 1992, p.185).

Qualquer sistema é instável e sujeito a alterações com o decorrer do tempo. A definição de diferentes tipos de objectos surge da necessidade de garantir estabilidade aos sistemas informáticos. A estabilidade de um sistema será maior se as alterações a efectuar forem bem localizadas. Sendo assim, pretende-se, por exemplo, que eventuais modificações no *interface* do utilizador com o sistema informático apenas vão afectar os objectos dessa categoria (objectos de *interface*).

O método contempla, mas de forma simples, a necessária identificação de subsistemas para projectos de média ou grande dimensão. A delimitação dos subsistemas deve ser realizada de acordo com as diferentes unidades organizacionais (*marketing*, produção, vendas etc.) (Jacobson *et al.*, 1992, p.190).

Apesar de Jacobson *et al.* (1992, p.133) afirmarem que o modelo de análise do Objectory tem como função representar o sistema de informação através de uma sólida e extensível estrutura de objectos, independentemente do actual ambiente de implementação, constatamos que, no desenvolvimento do processo de análise, o "objecto" em estudo não é o sistema organizacional mas o novo sistema computacional a conceber. A própria identificação de objectos de *interface* e objectos de controlo é sintomático desta afirmação.

A notação utilizada pelo Objectory é original, representando os objectos através de pequenos círculos, existindo algumas diferenças de pormenor que permitem diferenciar as várias categorias de objectos [11]. Os objectos são relacionados através de associações de

---

[11] Consultar a notação gráfica, em anexo, na pág. 147.

comunicação ou de estruturas de agregação. A notação utilizada devido à sua simplicidade apresenta algumas limitações.

Existe uma estreita relação entre as fases de análise e desenho, porque ambas se centram no novo sistema informático a desenvolver.

A estrutura de conceitos do Objectory é sólida e incorpora os princípios básicos relativos ao paradigma da orientação para objectos.

4.3 Avaliação dos métodos de análise orientada para objectos.

Não é uma tarefa fácil avaliar métodos substancialmente distintos quando a sua aplicação é diminuta e com resultados nem sempre amplamente divulgados, aumentando naturalmente o grau de subjectividade inerente à apreciação.

O nosso objectivo não é eleger um método de análise orientada para objectos, mas sim destacar alguns aspectos globalmente relevantes e evidenciar algumas lacunas existentes.

A matriz da fig. 4.1 apresenta em coluna os diferentes métodos abordados neste capítulo e em linha um conjunto de condições que, em nosso entender, contribuem para a adaptação de um método de análise de sistemas de informação orientado para objectos a sistemas de informação de gestão.

Essas condições, já referidas no início do capítulo, são:

a) Incorporação no método de uma filosofia de orientação para objectos.

b) Capacidade de representação organizacional das técnicas utilizadas no método.

c) Nível de integração da fase de análise no processo de desenvolvimento de *software*.

d) Capacidade de gestão da complexidade, através de mecanismos de decomposição funcional e interligação de subsistemas organizacionais.

e) Capacidade de o modelo resultante da fase de análise ser entendido pelos utilizadores do sistema de informação e responsáveis pela gestão, de forma a validarem a sua construção e contribuírem para o seu aperfeiçoamento.

f) Capacidade de identificar a informação necessária para os sistemas de informação de gestão, aos níveis estratégico, táctico e operacional.

A valorização apresentada na fig. 4.1 não permite uma classificação individual dos métodos de análise orientada para objectos. A importância das diversas alíneas definidas como requisitos é naturalmente diferente pelo que teriam de ser distintamente ponderadas. Acresce também o facto de existirem outros aspectos que poderão complementar essa eventual avaliação e que não são evidenciados neste trabalho. Aliás, pensamos que os métodos de análise não têm necessariamente de ser melhores ou piores. A escolha de um ou outro método poderá ser mais adequada consoante a formação e capacidade da equipa de desenvolvimento e a especificidade do sistema a desenvolver.

	AOO Coad/Yourdon	OMT Rumbaugh *et al.*	DBR Wirfs-Brock *et al.*	AOO Shlaer/Mellor	AOO Martin/Odell	SOM Velho	OBLOG Semadas *et al.*	Objectory Jacobson *et al.*
a)	5	5	6	2	4	6	6	5
b)	3	3	2	4	4	2	2	2
c)	4	4	4	3	4	4	4	4
d)	2	2	3	4	2	1	1	2
e)	3	3	3	3	4	2	2	2
f)	2	2	2	2	2	2	2	2

6 , 5 - Forte 4 , 3 - Médio / Moderado 2 , 1 - Fraco

Fig. 4.1 - Matriz de avaliação dos métodos de análise orientada para objectos.

A matriz da fig. 4.1 contribui graficamente para visualizar o posicionamento dos diferentes métodos, em relação às condições anteriormente expostas, e evidenciar alguns aspectos que vamos passar a enunciar:

1° (alínea a)) - Os métodos de análise, designados de orientados para objectos, têm diferentes níveis de absorção do paradigma da orientação para objectos.

Alguns, apesar de incorporarem conceitos deste paradigma, resultam nitidamente da evolução de métodos de análise estruturada. É o caso da análise orientada para objectos de Martin e Odell e, também, embora de uma forma menos acentuada, do OMT e do método de Coad e Yourdon.

Surgem igualmente métodos, fortemente orientados para objectos, que derivam da adopção dos princípios de orientação para objectos inicialmente introduzidos nas linguagens de programação e que procuram a construção de um modelo integrado que facilite a transposição da análise para a implementação, incluindo normalmente

uma linguagem de especificação de sistemas muito semelhante a uma linguagem de programação de computadores. Nestas abordagens, é notório a inexistência de diagramas de fluxos de dados ou outros afins (casos do OBLOG e do SOM).

Outros métodos, indiscutivelmente orientados para objectos, apresentam características relativamente peculiares, como acontece com o Desenho Baseado em Responsabilidades de Wirfs-Brock *et al.* e o Objectory.

Existem igualmente trabalhos que, eventualmente por razões comerciais ou devido à sua prematura data de publicação, se designam de orientados para objectos, quando, na verdade, não incluem os princípios de base deste novo paradigma. Neste caso situamos a abordagem de Shlaer e Mellor (1988).

2º (alínea b))- Uma parte significativa dos actuais métodos de análise orientada para objectos, principalmente aqueles que não têm raízes nos métodos de análise estruturada, centram-se na análise de um sistema informático. A fase de análise de sistemas de informação é submergida pelo desenho de um sistema computacional, não se reconhecendo devidamente a importância das especificações organizacionais reflectidas no sistema de informação.

A capacidade de representação organizacional diminui com a utilização de métodos de análise orientada para objectos. Mesmo os métodos que se concentram na análise do sistema de informação, como o método apresentado por Martin e Odell (1992), não transmitem uma boa perspectiva organizacional das especificações funcionais do sistema de informação.

As técnicas utilizadas pelos métodos de análise orientada para objectos são, de um modo geral, mais complexas do que as técnicas de análise estruturada. Essa complexidade resulta da própria essência do conceito de objecto (mais complexo do que o tradicional conceito de entidade) e naturalmente aumenta nos métodos que mais profundamente incorporam uma filosofia de orientação para objectos.

As técnicas de análise orientada para objectos apresentam-se actualmente insuficientemente dotadas de mecanismos de decomposição e interligação de subsistemas organizacionais.

3º (alínea c)) - No desenvolvimento orientado para objectos, a fase denominada de análise de sistemas está mais facilmente interligada com a fase de desenho. É usual analisar o sistema informático em detrimento do sistema de informação da organização. Sendo assim, ambas as fases têm o mesmo "objecto" pelo que a sua ligação é natural.

A interligação a montante da fase de análise com uma eventual fase de planeamento de sistemas de informação parece-nos mais problemática. Se já existia, em nosso entender, algum distanciamento entre estas duas fases, esse distanciamento aumenta na medida em que uma filosofia de concepção e desenvolvimento puramente orientada para objectos pressupõe uma acentuada redução da perspectiva funcional do sistema de informação.

Alguns métodos de planeamento de sistemas de informação, como por exemplo o B.S.P. (*Business Systems Planning*), centram-se na elaboração de uma arquitectura da informação, através da identificação e interligação de actividades e classes de dados, emergindo então os diversos subsistemas organizacionais. A derivação destes métodos de planeamento de sistemas de informação para a análise estruturada parece-nos mais fácil do que para métodos de análise orientados para objectos.

Apesar das actividades e entidades entendidas ao nível do planeamento estratégico serem mais abrangentes do que o conceitos sinónimos de análise de sistemas, era possível identificar alguma relação entre os dois níveis. Essa relação é mais ténue se a análise for desenvolvida numa perspectiva orientada para objectos.

4º (alínea d)) - A generalidade dos métodos apresenta insuficiências ao nível da gestão da complexidade, pelo que colocam em risco a sua aplicação a sistemas de informação organizacionais de acentuada dimensão. Os mecanismos introduzidos de partição e interligação de subsistemas são rudimentares e exiguamente descritos.

Os trabalhos de Shlaer e Mellor (1992) e de Wirfs-Brock *et al.* (1991) constituem excepções, contemplando técnicas de gestão da complexidade, embora seja discutível a orientação para objectos do trabalho de Shlaer e Mellor.

Uma solução possível para este problema (embora não seja em nosso entender a solução ideal) consiste na combinação de técnicas provenientes de diferentes métodos. Por exemplo, a técnica de partição de subsistemas apresentada por Wirfs-Brock poderia ser utilizada como um complemento ao SOM, ou ao OBLOG.

5º (alínea e)) - A concepção de métodos orientados para objectos foi direccionada numa óptica essencialmente técnica, para colmatar insuficiências de produtividade e qualidade sentidas pelos profissionais de desenvolvimento de *software*. Atribui-se pouca importância à necessidade de validação por parte dos utilizadores da representação do sistema de informação na organização.

6° (alínea f)) - Os métodos de análise orientada para objectos estão, na sua generalidade e no momento actual, vocacionados, não para analisar sistemas de informação, mas, fundamentalmente, para conceber sistemas informáticos.

O seu campo de aplicação centra-se principalmente na automatização de sistemas de informação de nível operacional, existindo evidentes dificuldades na sua aplicação aos níveis táctico e estratégico.

Os sistemas de informação tácticos e estratégicos utilizam informação proveniente dos sistemas de informação operacionais, mas incorporam igualmente informação específica das actividades tácticas e estratégicas. Na medida em que essa informação não possui características estruturadas é mais difícil o seu tratamento recorrendo às tecnologias da informação.

Métodos de análise de sistemas de informação profundamente orientados no sentido da concepção de um sistema informático, como acontece com a generalidade dos métodos de análise orientada para objectos, apresentam algumas dificuldades naturais em analisar informação táctica e estratégica[12] . É perfeitamente evidente que não foram concebidos com essa finalidade, mas subsiste o problema de supostamente se imaginar que desempenham funções de análise do sistema de informação de gestão quando, na verdade, este conceito é muito mais abrangente e importante, do que a simples automatização do sistema de informação operacional que o integra.

[12] Existem actualmente métodos, como, por exemplo, o *Soft Systems Methodology*, desenvolvido na Universidade de Lancaster por uma equipa liderada por Peter Checkland, que foram concebidos para analisar sistemas de informação aos diferentes níveis de gestão, principalmente aos níveis táctico e estratégico, onde subsiste um assinalável número de problemas não estruturados. Uma descrição deste método pode ser encontrada em CHECKLAND, Peter, (1981), *Systems Thinking, Systems Practice*, Chichester, John Wiley & Sons e em CHECKLAND, Peter e SCHOLES, Jim, (1990), *Soft Systems Methodology in Action*, Chichester, John Wiley & Sons.

Capítulo V

Conclusões

O paradigma da orientação para objectos nasce da necessidade de aumentar a qualidade e a produtividade no desenvolvimento de aplicações informáticas.

Inicialmente, esse paradigma surgiu incorporado nas linguagens de programação orientadas para objectos, com evidentes e comprovadas vantagens.

O facto da fase de programação depender dos resultados provenientes das fases de análise e desenho, resultou na tentativa de extensão, a estas duas fases, dos princípios fundamentais de orientação para objectos. Uma parte significativa dos métodos existentes de análise orientada para objectos resulta da incorporação destes princípios nos tradicionais métodos de análise estruturada. Outros métodos, fortemente orientados para objectos, procuram a adopção de um modelo integrado que possibilite acelerar o ciclo de desenvolvimento de *software*, através de uma maior integração entre as fases de análise, desenho e programação.

A maioria dos métodos analisados neste trabalho apresentam como objecto de estudo, não o sistema de informação da organização, mas sim, um sistema computacional de suporte àquele, pelo que podemos dizer que são métodos mais vocacionados para o desenho de sistemas informáticos do que para a análise das especificações do sistema de informação.

As excepções surgem nos métodos que evoluíram a partir da análise estruturada (como a análise orientada para objectos apresentada por Martin e Odell (1992) e o método de Coad e Yourdon (1991a)).

Os métodos que possuem técnicas de representação funcional (como por exemplo os DFDs) possuem alguma capacidade de especificação do sistema de informação organizacional, apesar de por vezes apresentarem uma tendência para se concentrarem na concepção de um sistema informático (como acontece com o OMT).

Um modelo do sistema de informação puramente orientado para objectos é mais complexo de conceber e de interpretar, porque implica a compreensão conjunta das perspectivas dinâmica e estrutural da informação, que estão conjuntamente capsuladas no seu conceito básico fundamental que é o conceito de objecto.

Um problema evidenciado na eficiência da generalidade dos métodos de análise orientada para objectos é o facto de estes apresentarem mecanismos muito rudimentares de gestão da complexidade, existindo uma evidente necessidade de os dotar de técnicas de partição e interligação de subsistemas organizacionais, para que seja possível a sua utilização em projectos de elevada dimensão e complexidade.

Concordamos que a elaboração de diagramas de fluxos de dados não é coerente com uma abordagem de análise orientada para objectos. Na medida em que o objecto é definido como um elemento que incorpora conjuntamente dados e processos, será, logicamente, conveniente dispor de uma técnica que represente simultaneamente dados e processos que, portanto, represente objectos. Este facto apresenta um problema associado. Os diagramas de fluxos de dados são uma técnica inata de decomposição funcional, permitindo a partição e interligação de subsistemas, possibilitando uma visão global e integrada, mas simultaneamente detalhada, de sistemas complexos constituídos por inúmeras estruturas de dados e processos, o que facilita a abordagem de análise e a gestão do próprio projecto de informatização. A não concepção de diagramas de fluxos de dados, ou inexistência de uma técnica alternativa, levanta obviamente o problema da subdivisão do sistema organizacional de forma a permitir o seu estudo e representação.

Os actuais métodos de análise orientada para objectos adaptam-se melhor à informatização de pequenos sistemas que estejam basicamente dependentes da construção de aplicações relativamente limitadas em número e extensão. Para a automatização de sistemas de informação de gestão, em organizações de elevada dimensão e complexidade, onde existe uma enredada teia de subsistemas, é fundamental uma análise detalhada do sistema de informação numa perspectiva organizacional e a utilização de técnicas que permitam a divisão e interligação desses subsistemas.

Tal como acontece com os métodos de análise estruturada, os métodos de análise orientada para objectos aplicam-se basicamente aos sistemas de informação de gestão de nível operacional, não incluindo uma análise táctica e estratégica da informação necessária ao funcionamento da organização. Na verdade, não foram concebidos com essa finalidade, e não pretendemos afirmar que se devem alterar no sentido de ser viável a sua aplicação aos níveis táctico e estratégico, pensamos contudo que não deve existir a pretensão de que é possível, recorrendo a métodos de análise orientada para objectos, analisar a globalidade do sistema de informação de gestão.

A análise do sistema de informação de nível táctico e estratégico envolve, por natureza, a necessidade de utilizar técnicas com características substancialmente diferentes das existentes nos actuais métodos de análise de sistemas de informação.

O entendimento da informação numa perspectiva táctica e estratégica implica a utilização de modelos de gestão, construídos por quem conhece profundamente a natureza do "negócio", de modo a definir quais as necessidades e importância da informação para o sucesso empresarial e quais os modelos de decisão adequados ao funcionamento da organização. Obviamente que métodos pragmáticos de desenvolvimento de aplicações, como os métodos actuais designados de análise de sistemas de informação, quer sejam orientados ou não para objectos, não cobrem estes requisitos.

A influência da tecnologia orientada para objectos nos métodos de análise de sistemas, realçando a perspectiva de construção de sistemas computacionais em detrimento das especificações organizacionais dos sistemas de informação, vem ao encontro da distinção entre a essência dos sistemas de informação e a utilização de tecnologia informática.

Os métodos de análise de sistemas de informação orientados para objectos nascem da necessidade de resolver problemas evidenciados ao nível da "oferta" dos sistemas informáticos. Com o objectivo de acelerar a produtividade e aumentar a eficiência dos sistemas informáticos, subvalorizou-se as especificações da "procura" nos sistemas de informação.

O planeamento estratégico de sistemas de informação e a análise de sistemas de informação devem ser estudados numa perspectiva organizacional, porque relacionam-se com a procura de soluções para os sistemas de informação nas organizações.

A engenharia informática e as tecnologias da informação são responsáveis pelo paradigma da oferta. A sua função corresponde à construção de sistemas computacionais que satisfaçam as necessidades da procura.

A constatação desta realidade, concretizada através de uma forte relação entre os paradigmas da procura e da oferta, é importante para que os sistemas de informação nas organizações sejam automatizados de acordo com as necessidades de gestão e os requisitos organizacionais, utilizando para a sua automatização métodos eficientes de concepção e desenvolvimento de sistemas informáticos. Neste segundo ponto, os métodos orientados para objectos contribuem de forma significativa mas, para que sejam aplicáveis numa perspectiva organizacional, é importante o seu enquadramento a montante com métodos de planeamento estratégico e de análise das especificações do sistema de informação.

BIBLIOGRAFIA
CONSULTADA

ANDLEIGH, Prabhat K. e GRETZINGER, Michael R., (1992), *Distributed Object-Oriented Systems Analysis*, Englewood Cliffs, Prentice Hall.

ANSOFF, H. Igor, (1977), *Estratégia Empresarial*, S.Paulo, McGraw-Hill (tradução do original: *Corporate Strategy*, New York, McGraw-Hill, 1965).

ANTHONY, Robert N., (1981), *Planning and Control Systems: A Framework for Analysis*, 11ª edição (1ª edição 1965), Boston, Harvard University Press.

ARNOLD, Patrick, BODOFF, Stephanie, COLEMAN, Derek, GILCHRIST, Helena e HAYES, Fiona, (1991), *An evaluation of five Object-Oriented Development Methods*, Bristol, HP Laboratories.

AVISON, D.E. e FITZGERALD, G., (1988), *Information Systems Development: Methodologies, Techniques and Tools*, Oxford, Blackwell Scientific Publications.

BAILIN, Sidney C., (Maio 1989), «An Object-Oriented Requirements Specification Method», *Communications of the ACM*, vol. 32, nº 5, pp. 608-623.

BARRY, Douglas K., (Jan./Fev. 1993), «ODBMS Feature Listing», *Object Magazine*, vol. 2, nº 5, pp. 48-53.

BECK, K. e CUNNINGHAM, W., (1989), «A Laboratory For Teaching Object-Oriented Thinking», *Proceedings OOPSLA '89 - Conference on Object-Oriented Programming, Systems, Languages and Applications*, pp. 1-6.

BERARD, E.V., (1989), «Object-Oriented Requirements Analysis and Design», *Proceedings of the Sixth Washington Ada Symposium*, pp. 9-11.

BERTALANFFY, Ludwing Von, (1973), *Teoria Geral dos Sistemas*, Petrópolis, Editora Vozes Lda. (tradução do original: *General Systems Theory*, Editora George Braziller, 1968).

BERTINO, Elisa e MARTINO, Lorenzo, (Abril 1991), «Object-Oriented Database Management Systems: Concepts and Issues», *IEEE Computer*, vol. 24, nº 4, pp. 33-47.

BLAHA, Michael R., PREMERLANI, William J. e RUMBAUGH, James E., (Abril 1988), «Relational Database Design Using an Object-Oriented Methodology», *Communications of the ACM*, vol. 31, nº 4, pp. 414-427.

BLOOR, Robin, (Julho 1992), «The End of Relational?», *DBMS*, vol. 5, nº 8, pp. 8-10.

BOEHM, Barry W., (1981), *Software Engineering Economics*, Englewood Cliffs, Prentice Hall.

BOEHM, Barry W., (Maio 1988), «A Spiral Model of Software Development and Enhancement», *IEEE Computer*, vol. 21, nº 5, pp. 61-72.

BONAR, Jeffrey, (Outubro 1990), «Crown jewels: the value of object-oriented technology in MIS departments», *Hotline on Object-Oriented Technology*, vol. 1 , nº 12 , pp. 1,3-5.

BOOCH, G., (1991), *Object-Oriented Design With Aplications*, Redwood City, Benjamin/Cummings.

BOULDING, Kenneth E., (Abril 1956), «General Systems Theory. The Skeleton of Science», *Management Science*, pp. 197-208.

CATTELL, Roderic G., (1991), *Object Data Management*, Reading, Addison-Wesley.

CHECKLAND, Peter, (1981), *Systems Thinking, Systems Practice*, Chichester, John Wiley & Sons.

CHECKLAND, Peter e SCHOLES, Jim, (1990), *Soft Systems Methodology in Action*, Chichester, John Wiley & Sons.

CHEN, P., (Março 1976), «The Entity-Relationship Model - Towards a Unified View of Data», *ACM Transactions on Database Systems*, vol. 1, nº 1, pp. 9-36.

CHIAVENATO, Idalberto, (1985), *Administração: Teoria, Processo e Prática*, S. Paulo, McGraw-Hill.

COAD, Peter e YOURDON, Edward, (1991a), *Object-Oriented Analysis*, 2ª edição (1ª edição 1990), Englewood Cliffs, Prentice Hall.

COAD, Peter e YOURDON, Edward, (1991b), *Object-Oriented Design*, Englewood Cliffs, Prentice Hall.

COAD, Peter, (1992) , «Object-Oriented Patterns», *Communications of the ACM*, vol. 35, nº 9, pp. 152-159.

CODD, E.F., (Junho 1970), «A Relational Model of Data for Large Shared Data Banks», *Communications of the ACM*, vol. 13, nº 6, pp. 377-387.

CODD, E.F., (Dezembro 1979), «Extending the Database Relational Model to Capture More Meaning», *ACM Transactions on Database Systems*, vol. 4, nº 4, pp. 397-434.

CODD, E. F., (Outubro 1985), «Is Your DBMS Really Relational ?», *Computer World*.

COELHO, Helder, (1986), *Tecnologias da Informação: Sistemas inteligentes, perspectivas, possibilidades e implicações*, Lisboa, Dom Quixote.

COMISSÃO TÉCNICA DE NORMALIZAÇÃO DA TECNOLOGIA INFORMÁTICA - CT113, (1991), *Glossário de Termos Informáticos*, Lisboa, Instituto de Informática.

COX, Brad J., (1986), *Object-Oriented Programming - An Evolutionary Approch*, Reading, Addison-Wesley.

CRIBBS, John, MOON, Suzanne e ROE, Collen, (1992), *An Evaluation of Object-Oriented Analysis and Design Methodologies*, New York, SIGS Books.

DANIELS, A. e YEATS D.A., (1971), *Basic Training in Systems Analysis*, 2ª edição, Londres, Pitman.

DATE, C.J., (1984), *An Introdution to Database Systems*, Volume 2, 2ª edição (1ª edição 1983), Reading, Addison-Wesley.

DATE, C.J., (1990), *An Introdution to Database Systems*, Volume 1, 5ª edição (1ª edição 1975), Reading, Addison-Wesley.

DE CHAMPEAUX, Dennis e FAURE, Penelope, (Mar./Abr. 1992), «A comparative study of object-oriented analysis methods», *Journal of Object-Oriented Programming*, vol. 4, nº10, pp. 21-33.

DE MARCO, Tom, (1978), *Structured Analysis and System Specifications*, Englewood Cliffs, Prentice Hall.

DOS SANTOS, António Mendes, FERNANDES, José Palma e DA CUNHA, Luís Arriaga, (1986), «Desenvolvimento expedito de aplicações», *Comunicação do 4º Congresso Português de Informática*, Lisboa.

DOWNS, E., CLARE, Peter e COE, Ian, (1992), *Structured Systems Analysis and Design Method: Application and Context*, 2ª edição (1ª edição 1988), Hemel Hempstead, Prentice Hall.

EDELSTEIN, Herb, (Novembro 1991), «Relational vs. Object-Oriented», *DBMS*, vol. 4, nº 12, pp. 68-79.

FERREIRA, Rogério Fernandes, (1985), *Lições de Gestão Financeira*, Volume I, Livraria Arnaldo, Coimbra.

FISHER, Alan S., (1988), *CASE: Using Software Development Tools*, New York, John Wiley & Sons.

FOWLER, Martin, (1992), «A Comparation of Object-Oriented Analysis and Design Methods», *Tuturial for OOPSLA92 - Conference on Object-Oriented Programming, Systems, Languages and Applications.*

GALACSI, (1986), *Les Systèmes d'Information, Analyse et Conception*, Paris, Dunod Informatique.

GANE, Chris e SARSON, Trish, (1979), *Structured Systems Analysis: Tools and Techniques*, Englewood Cliffs, Prentice Hall.

GIBSON, Elizabeth, (Outubro 1990), «Object - Born and Bred», *Byte*, pp. 245-252.

GOLDBERG, A. e ROBSON, (1983), D., Smalltalk 80: *The language and it's implementation*, Reading, Addison-Wesley.

GOOR, Geert van der, HONG, Shuguang e BRINKKEMPER, Sjaak, (1992), *A Comparison of Six Object-Oriented Analysis and Design Methods*, Enschede (Netherlands), University of Twente.

GORRY, G. Anthony e SCOTT-MORTON, Michael S., (1971), «A Framework for Management Information Systems», *Sloan Management Review (M.I.T.)*, vol.13, nº 1, pp.55-70.

GRADY, Robert B. e CASWELL, Deborah L., (1987), *Software Metrics: establishing a company-wide program*, Englewood Cliffs, Prentice Hall.

GRAHAM, Ian, (1991), *Object-Oriented Methods*, Wokingham, Addison-Wesley.

GUTON, Tony, (1990), *Inside Information Technology: A Pratical Guide to Management Issues*, Hemel Hempstead, Prentice Hall.

HAMPTON, David R., (1983), *Administração Contemporânea*, 2ª edição (1ª edição 1981), São Paulo, McGraw-Hill.

HENDERSON-SELLERS, Brian e EDWARDS, Julian M., (Setembro 1990), «Object-Oriented Systems Life Cycle», *Communications of the ACM*, vol. 33, nº 9, pp. 142-59.

HENDERSON-SELLERS, Brian, (1992), *A Book of Object-Oriented Knowledge*, Englewood Cliffs, Prentice Hall.

HUET, Gérard e ROUSSET, Jean, (1980), *Systèmes d'information*, Paris, Sirey.

HUGHES, John G., (1991), *Object-Oriented Databases*, Englewood Cliffs, Prentice Hall.

INSTITUTO DE LINGUÍSTICA TEÓRICA E COMPUTACIONAL, (1993), *Dicionário de Termos Informáticos*, Lisboa, Edições Cosmos.

JACOBSON, Jan, CHRISTERSON, Magnus, JONSSON, Patrik e ÖVERGAARD, Gunnar, (1992), *Object-Oriented Software Engineering*, Wokingham, ACM Press/Addison-Wesley.

JONES, Carpers, (1986), *Programming Productivity*, New York, McGraw-Hill.

KHOSHAFIAN, Setrag e ABNOUS, Razmik, (1990), *Object-Orientation: Concepts, Languages, Databases, User Interfaces*, New York, John Wiley & Sons.

KORSON, Tim, MCGREGOR, John D., (Setembro 1990), «Understanding Object-Oriented: A Unifying Paradigm», *Communications of the ACM*, vol. 33, nº 9, pp. 40-60.

LEE, Sangbum e CARVER, Doris, (Janeiro 1991), «Object-oriented analysis and specification: a knowledge base approach», *Journal of Object-Oriented Programming*, vol. 3, nº 5, pp. 5-43.

LE MOIGNE, Jean Louis, (1974), *les systemes de décision dans les organisations*, Paris, Presses Universitaires de France.

LE MOIGNE, Jean Louis, (Nov. 1978a), «la théorie du système d'information organisationnel», *Informatique et Gestion*, Nº 101 , pp. 39-42.

LE MOIGNE, Jean Louis, (Dez. 1978b), «la théorie du système d'information organisationnel», *Informatique et Gestion*, N° 102, pp. 28-31.

LE MOIGNE, Jean Louis, (Jan./Fev. 1979a), «la théorie du système d'information organisationnel», *Informatique et Gestion*, N° 103, pp. 31-35.

LE MOIGNE, Jean Louis, (Março 1979b), «la théorie du système d'information organisationnel», *Informatique et Gestion*, N° 104, pp. 31-43.

LUCAS Jr., Henry C., (1986), *Information Systems Concepts for Management*, 3ª edição (1ª edição 1978), Singapura, McGraw-Hill.

MARCELINO, Henrique, (Dezembro 1990), «Planeamento estratégico para a renovação organizacional através das técnologias da informação», *Informação & Informática*, ano IV, n° 7, Lisboa, Instituto de Informática, pp. 7-13.

MARTIN, James, (1986a), *Information Engineering*, Volume 1, Carnforth, Savant.

MARTIN, James, (1986b), *Information Engineering*, Volume 2, Carnforth, Savant.

MARTIN, James, (1986c), *Information Engineering*, Volume 3, Carnforth, Savant.

MARTIN, James e McCLURE, Carma, (1983), *Software Maintenance: The problem and its solutions*, Englewood Cliffs, Prentice Hall.

MARTIN, James e ODELL James, (1992), *Object-Oriented Analysis and Design*, Englewood Cliffs, Prentice Hall.

McMENAMIN, Steve e PALMER, John, (1984), *Essential Systems Analysis*, Englewood Cliffs, Prentice Hall.

MEYER, Bertrand, (1988), *Object-Oriented Software Construction*, Hemel Hempstead, Prentice Hall.

MONARCHI, David E. e PUHR, Gretchen I., (Setembro 1992), «A Research Typology for Object-Oriented Analysis and Design», *Communications of the ACM*, vol. 35, n° 9, pp. 35-47.

NELSON, Jean-Marc, (Setembro 1992), «Applying Object-Oriented Analysis and Design», *Communications of the ACM*, vol. 35, n° 9, pp. 63-74.

NEUHOLD, Erich e STONEBRAKER, Michael, (Eds), (1988), «Future Directions in DBMS Research», *Proceedings of the Laguna Beach Participants*, International Computer Science Institute, University of California.

NOLAN, Richard L., (Mar./Abr. 1979), «Managing the crisis in data processing», *Harvard Business Review*, vol. 57, nº 2, pp. 115-126.

PAGE-JONES, Meilir, (Setembro 1992), «Comparing Techniques by Means of Encapsulation and Connascence», *Communications of the ACM*, vol. 35, nº 9, pp.147-151.

PALMER, John, MEYER, Bertrand, WEISS, Steven, PAGE-JONES, Meilir e CONSTANTINE, Larry L., (Nov./Dez. 1991), «Evolution *vs* Revolution: Should Structured Methods Be Objectified ?», *Object Magazine*, vol. 1, nº 4, pp. 30-41.

PARSAYE, Kamran, CHIGNELL, Mark, KHOSHAFIAN, Setrag e WONG, Harry, (1989), *Intelligent Databases: Object-Oriented, Deductive, Hipermedia Technologies*, New York , John Wiley & Sons.

PARSONS, Talcott, (1960), *Structure and Process in Modern Societies*, New York, The Free Press.

PELLAUMAIL, Phillipe, (1986), *La Méthode Axial*, Tomo I, Paris, Les editions d'organization.

PORTER, Michael E. e MILLAR, Victor E., (Jul./Ago. 1985), «How information gives you competitive advantage», *Harvard Business Review*, vol. 63, nº 4, pp. 149-160.

PORTER, Michael E. e MILLAR, Victor E., (Primavera 1986), «Pour battre vos concurrents, maitrisez mieux l'information», *Harvard-L'Expansion*, pp. 6-20.

REIX, Robert, (1971a), *L'Analyse en Informatique de Gestion*, Volume 1, Paris, Dunod.

REIX, Robert, (1971b), *L'Analyse en Informatique de Gestion*, Volume 2, Paris, Dunod.

RESENDE, P. e SILVA, A. R., (1990), «A Formal Object-Oriented Approach to Systems Specification: Package Router Revisited», Lisboa, INESC.

REYNOLDS, George W., (1988), *Information Systems for Managers*, St. Paul, West Publishing Company.

RIVAS, Filipe Gómez-Pallete, (1984), *Estruturas Organizativas e Information en la Empresa*, Madrid, Association para el Progreso de la Direction.

ROSS, D. T., (Janeiro 1977), «Structured analysis: A language for communicating ideas», IEEE *Transaction on Software Engineering*, vol. SE 3, nº1, pp. 16-34.

ROYCE, Winston W., (Agosto 1970), «Managing the Development of Large Software Systems», *IEEE Proceedings*, pp. 1-9.

RUMBAUGH, James, BLAHA, Michael, PERMERLANY, William, EDDY, Frederick e LORENSEN, William, (1991), *Object-Oriented Modeling and Design*, Englewood Cliffs, Prentice Hall.

SERNADAS, Amílcar *et al.*, (1989a), *OBLOG: Desenvolvimento de Sistemas com Objectos*, Lisboa, INESC.

SERNADAS, Amílcar, SERNADAS, Cristina e EHRICH, H., (1989b), *Object-Oriented Language Features for Information Systems Specification*, OBLOG Research Report, Lisboa, INESC.

SERNADAS, Amílcar, COSTA, José Felix e SERNADAS, Cristina, (1992), *Especificação de Objectos com Diagramas: Abordagem OBLOG*, Lisboa, INESC.

SERNADAS, Amílcar, (1993), «Abordagem OBLOG», *Informação & Informática*, nº 11, ano VII, Lisboa, Instituto de Informática, pp.4-7.

SHALAER, Sally e MELLOR, Steve, (1988), *Object-Oriented Systems Analysis*, Englewood Cliffs, Prentice Hall.

SHALAER, Sally e MELLOR, Steve, (1992), *Object Life Cycles: Modeling the World in States*, Englewood Cliffs, Prentice Hall.

SHEMER, Itzhak, (Junho 1987), «Systems Analysis: A Systemic Analysis of a Conceptual Model», *Comunications of the ACM*, vol. 30, nº 6, pp. 506-512.

SIMON, Herbert, (1960), *The New Science of Management Decision*, New York, Harper & Row.

SMITH, John Miles e SMITH, Diane C., (Março 1977), «Data Abstractions: Aggregation e Generalization», *ACM Transactions on Database Systems*, vol. 2, nº 2, pp. 105-133.

STOECKLIN, S.E., ADAMS, E.J. e SMITH, S., (1988), «Object-Oriented Analysis», *Proceedings of the Fifth Washington Ada Symposium*, Association for Computing Machinary, pp. 133-138.

STONEBRAKER, Michael, (1986), «Inclusion of New Types in Relational Database Systems», IEEE *in* STONEBRAKER, Michael (Eds.), 1988, *Readings in Database Systems*, San Mateo, Morgan Kaufmann Publishers, pp. 480-487.

STONEBRAKER, Michael, (1991), «Object Management in Postgres using Procedures» *in On Object-Oriented Database Systems*, New York, Spring-Verlag, pp. 53-64.

STROUSTRUP, B., (1986), *The C++ Programming Language*, Reading, Addison-Wesley.

SWANSON, E. B., (1976), «The Dimensions of Maintenance», *Proceedings of the Second International Conference on Software Engineering* , pp. 492-497.

TARDIEU, Hubert, ROCHFELD, Arnold e COLLETTI, René, (1984), *La Méthode MERISE: principes et outils*, Paris, Les éditions d'organization.

TAYLOR, David A., (1991a), *Object-Oriented Technology: A Managers Guide*, Reading, Addison-Wesley.

TAYLOR, David A., (Nov./Dez. 1991b), «Redefine analysis & design», *Object Magazine*, vol. 1 nº 4, pp. 16-18.

TAYLOR, David A., (1992a), *Object-Oriented Information Systems: Planning and Implementation*, New York, John Wiley & Sons.

TAYLOR, David A., (Set./Out. 1992b), «The coming convergence of object and relational databases», *Object Magazine*, vo. 2, nº 3, pp. 16-18.

TAYLOR, David A., (Jan./Fev. 1993), «A development lifecycle for object technology», *Object Magazine*, vol. 2, nº 5 , pp. 18-20, 24.

TEOREY, Toby J., YANG, Dongking e FRY, James P., (Junho 1986), «A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model», *Computing Surveys*, vol. 18, nº 2, pp. 197-222.

THOMSETT, R., (Outubro 1990), «Management implications of object-oriented development», *ACS Newsletter*, pp. 5-12.

TRACS, Will, (1988), *Software Reuse: Emerging Technology*, Washington, The Computer Society Press of the IEEE.

TSICHRITZ, D. C., e LOCHOVSKY, F.H., (1982), *Data Models*, Englewood Cliffs, Prentice Hall.

VELHO, Amândio Vaz, (1991), *SOM: A Semantic Object Model - A Motivação, os Conceitos e as Linguagens*, (versão provisória, cedida por gentileza do autor), Lisboa, INESC.

VELHO, Amândio Vaz, (1992), *Atributos em SOM*, (versão provisória, cedida por gentileza do autor), Lisboa, INESC.

WARD, John, (1990), *Strategic Planning for Information Systems*, West Sussex, John Wiley & Sons.

WINBLAD, Ann, EDWARDS, Samuel e KING, David, (1993), *Software Orientado ao Objecto*, Rio de Janeiro, Makron Books (tradução do original: *Object-Oriented Software*, Addison-Wesley, 1990).

WINSBERG, Paul e RICHARDS, Daniel, (Primav./Verão 1991), «Object-Oriented Analysis, Books and Tools», *InfoDB*, pp. 27-36.

WIRFS-BROCK, Rebecca, WILKERSON, Brian e WIENER, Lauren, (1990), *Designing Object-Oriented Software*, Englewood Cliffs, Prentice Hall.

YADAV, Surya B., BRAVOCO, Ralph R., CHATFIELD, Akemi T. e RAJKUMA, T.M., (Setembro 1988), «Comparison of Analysis Techniques for Information Requirement Determination», *Communications of the ACM*, vol. 31, nº 9, pp. 1090-1097.

YOURDON, Edward, (1986), *Managing the Structured Techniques*, Englewood Cliffs, Prentice Hall.

YOURDON, Edward, (1988), *Managing the System Life Cycle*, 2ª edição, Englewood Cliffs, Prentice Hall.

YOURDON, Edward, (1989), *Modern Structured Analysis*, 3ª edição, Englewood Cliffs, Prentice Hall.

ZORRINHO, Carlos, (1991), *Gestão da Informação*, Lisboa, Editorial Presença.

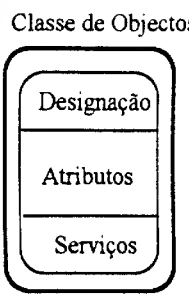
ANEXOS

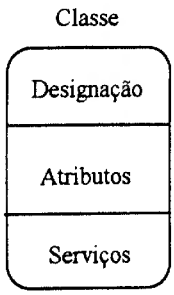
Resumo da notação gráfica utilizada pelos métodos de análise orientada para objectos.

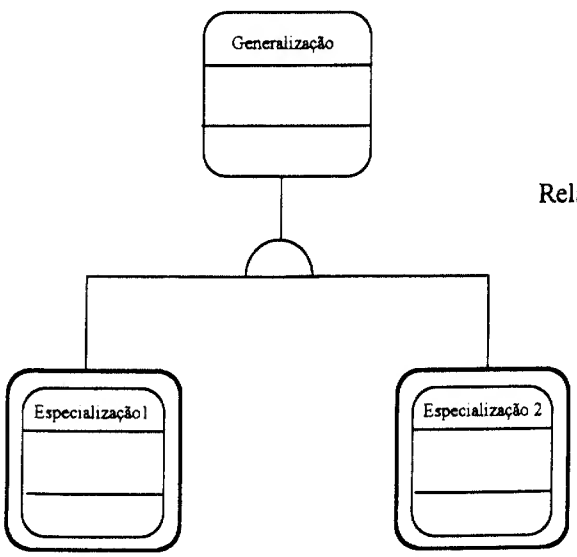
Análise Orientada para Objectos

Coad e Yourdon

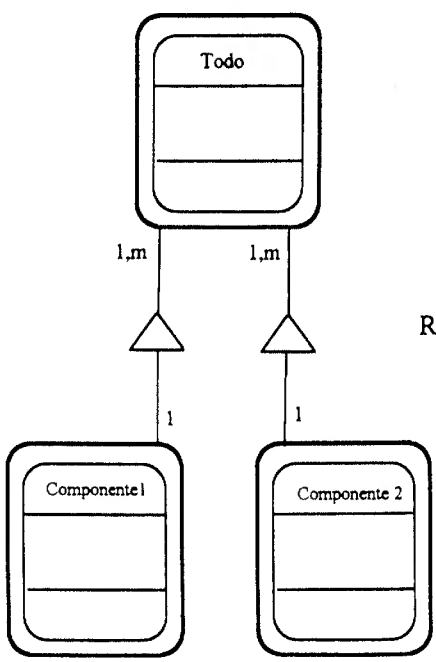
Diagrama de Classes & Objectos



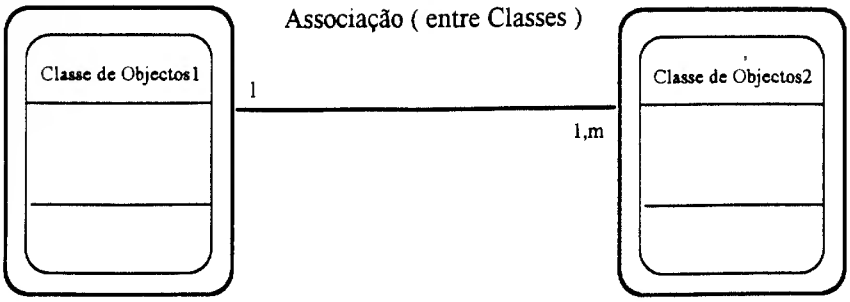




Relação de Generalização - Especialização



Relação de Agregação ("Todo-Parte")


Associação (entre Classes)
Classe de Objectos1
1
1,m
Classe de Objectos2

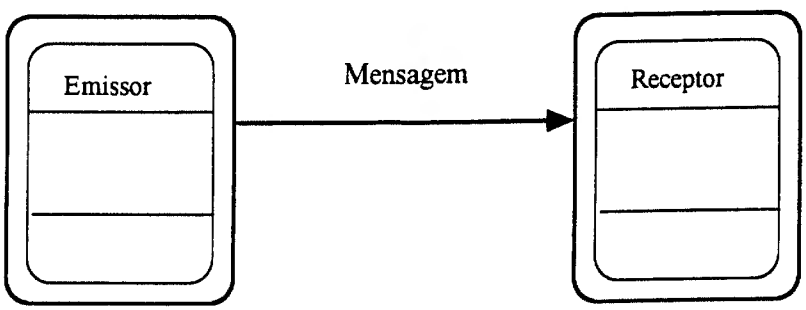

Emissor
Mensagem
Receptor

Diagrama de Estado do Objecto

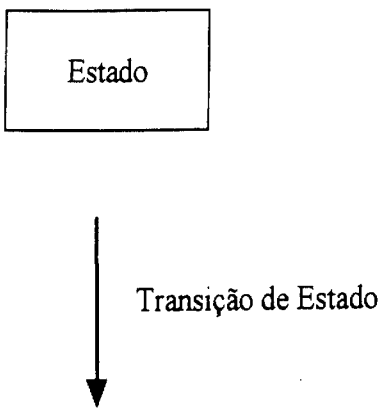
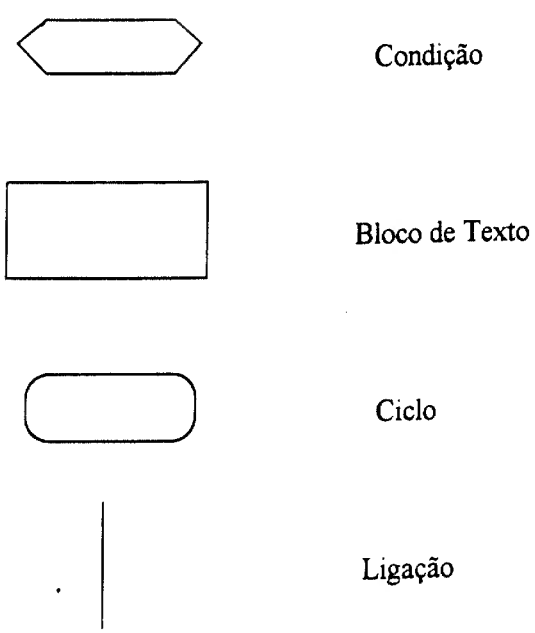


Diagrama de Serviços



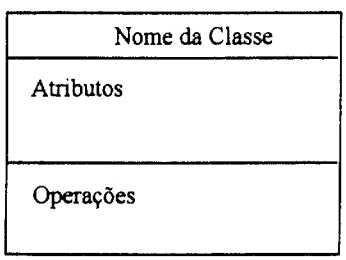
Object Modelling Technique

Rumbaugh *et al.*

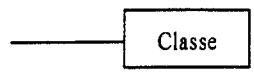
Modelo de Objectos

Classe

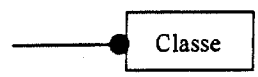
(notação simbólica) Nome da Classe



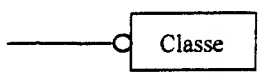
Cardinalidade da Associação



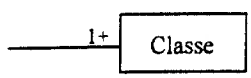
Exactamente um



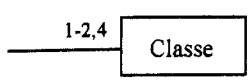
Vários (um ou mais)



Opcional (Zero ou um)

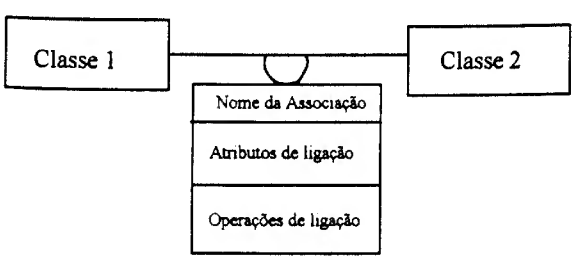


Um ou mais

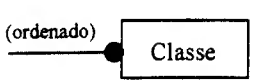


Numericamente especificado

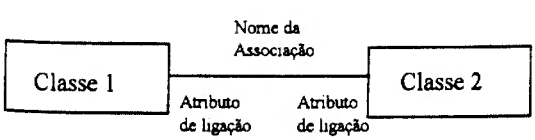
Classe Associativa



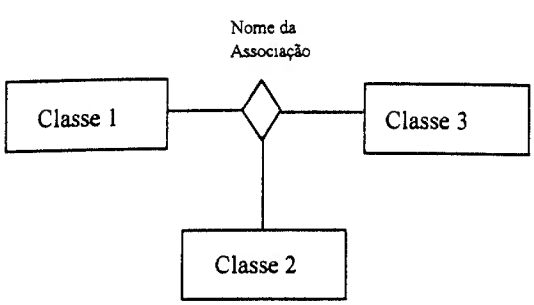
Ordenação



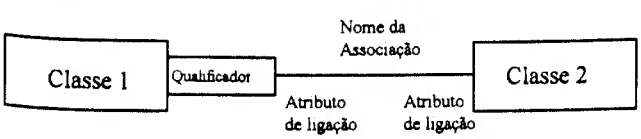
Associação



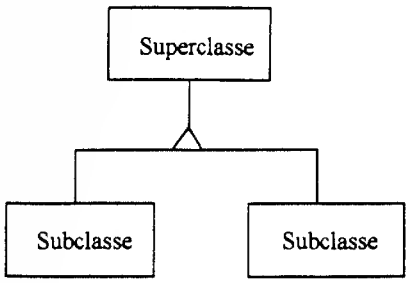
Associação ternária

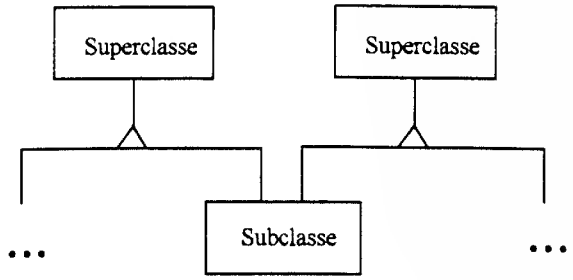


Associação Qualificada

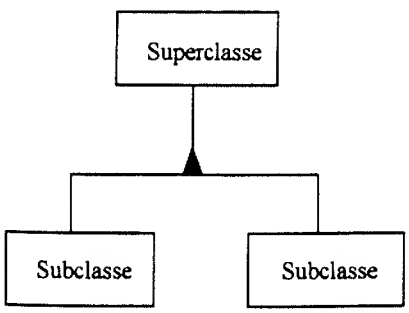


Generalização (Hereditariedade)

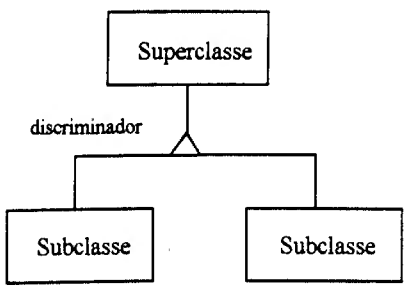




Múltipla Hereditariedade na Generalização

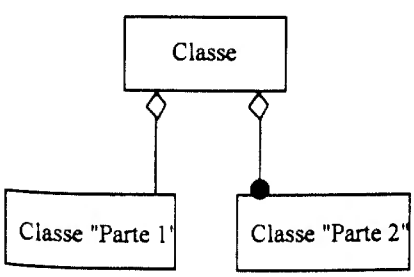


Subclasses com objectos comuns, com membros não disjuntos.

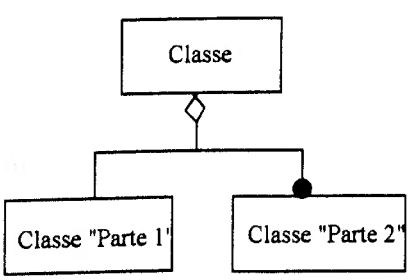


O discriminador é um atributo cujo valor diferencia as subclasses.

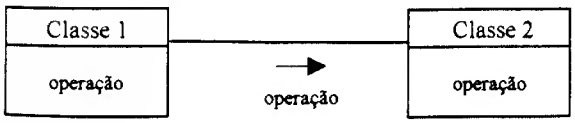
Agregação



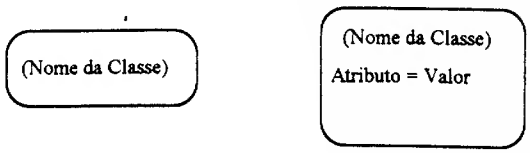
Agregação (notação alternativa)



Propagação de operações

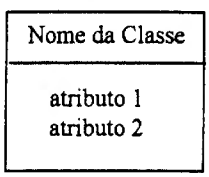


Instâncias dos objectos



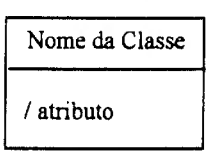
Relação de instância

Restrições em objectos

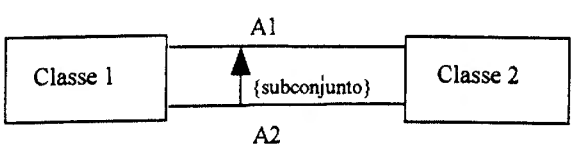


(atributo 1 > 0)

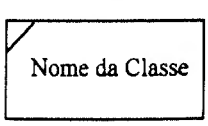
Atributo derivado



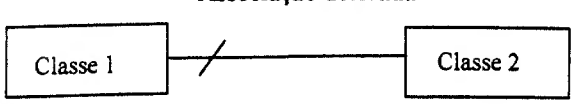
Restrições entre associações



Classe derivada

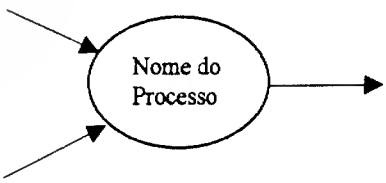


Associação derivada

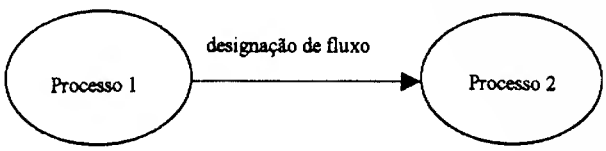


Modelo Funcional

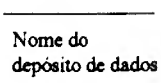
Processo:



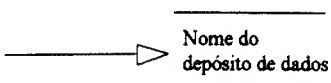
Fluxo de dados entre processos:



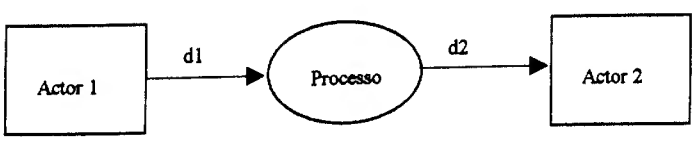
Depósito de dados:



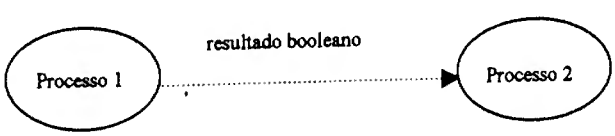
Fluxo de dados que resulta num depósito de dados:



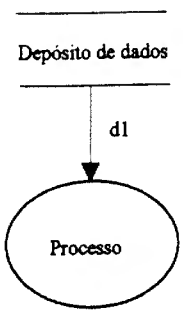
Actores:



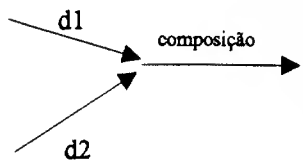
Fluxo de controlo:



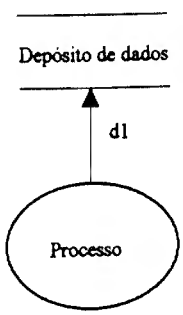
Acesso a um depósito de dados:



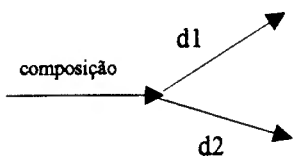
Composição de dados:



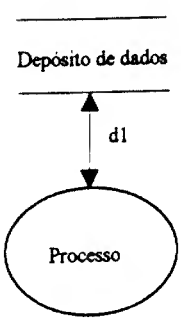
Actualização de um depósito de dados:



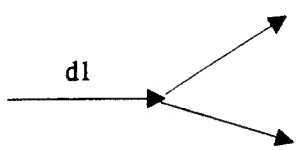
Decomposição de dados:



Consulta e actualização de um depósito de dados:

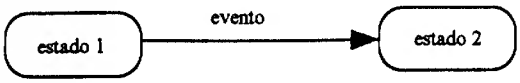


Duplicação de dados:

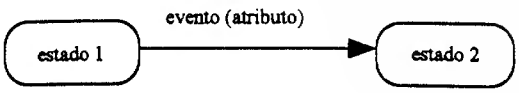


Modelo Dinâmico

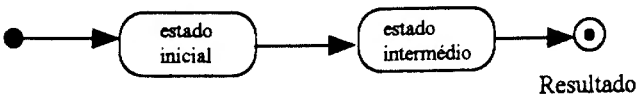
Transição entre estados através de um evento:



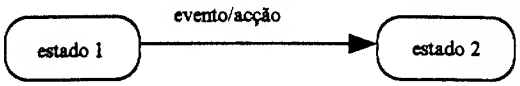
Eventos com atributos:



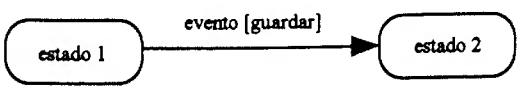
Estados inicial e final:



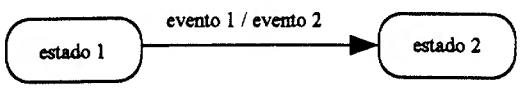
Acção realizada durante uma transição de estados:



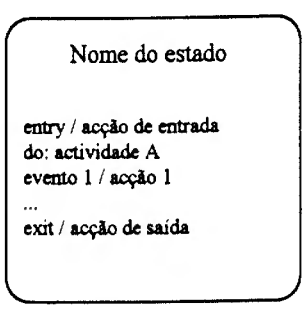
Transição guardada:



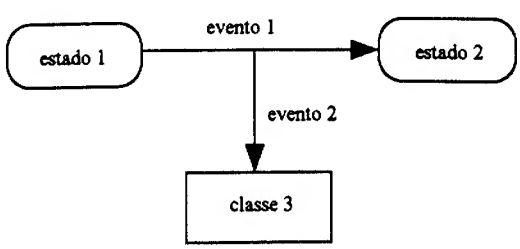
Evento de saída numa transição:



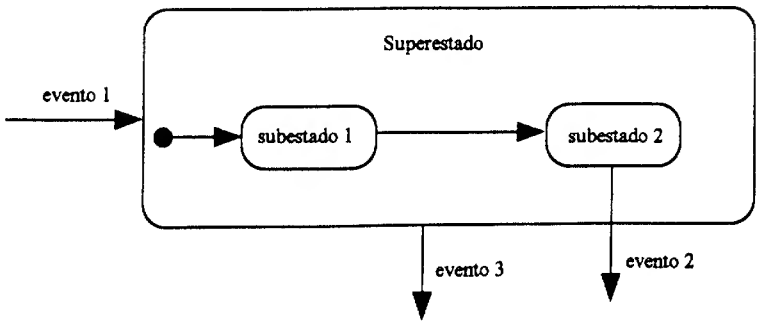
Estado incluindo actividades e acções:



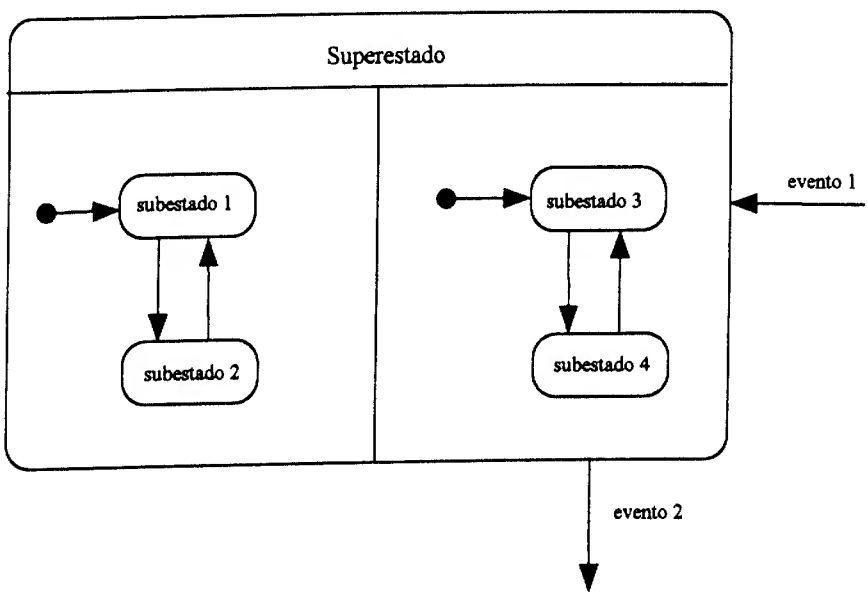
Envio de um evento para outra classe de objectos:



Generalização de estados:

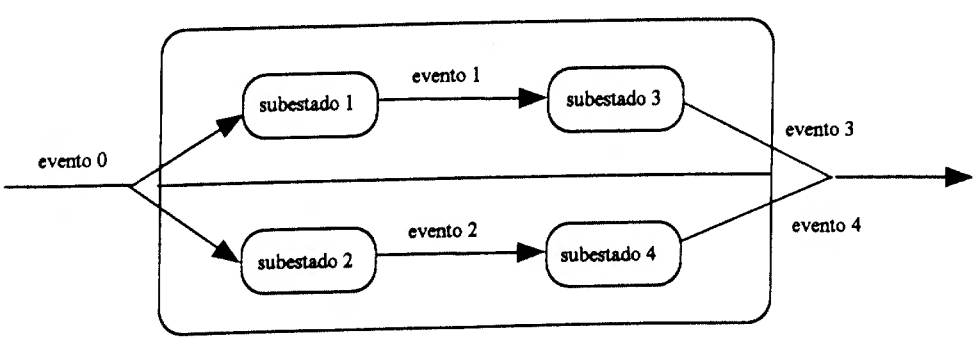


Subdiagrama concorrente:



Separação de controlo:

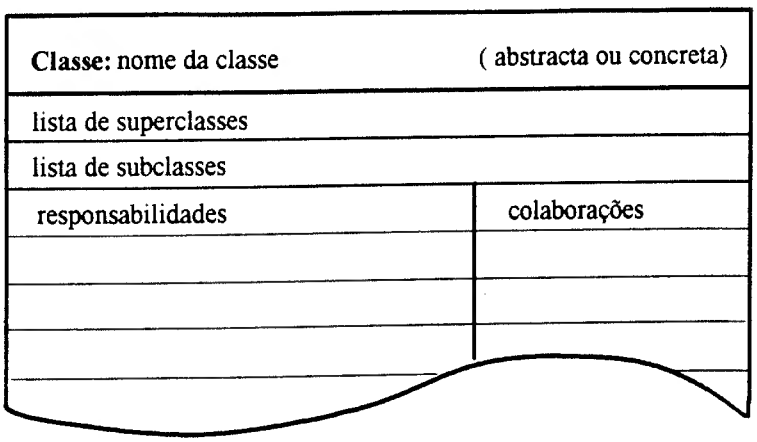
Sincronização de controlo:



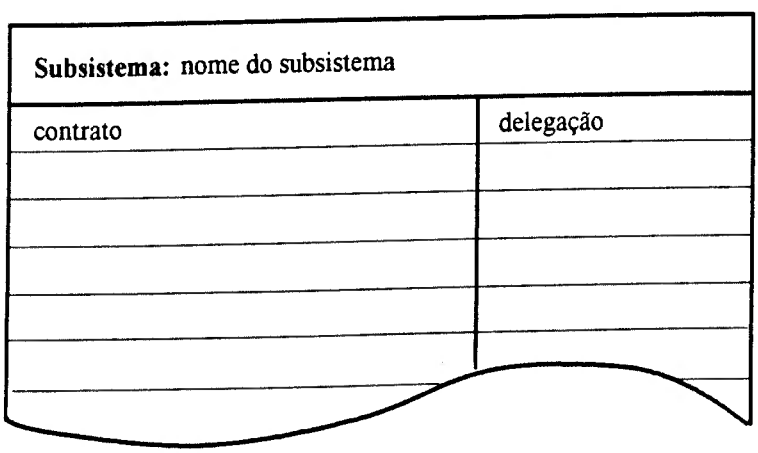
Desenho Baseado em Responsabilidades

Wirfs-Brock *et al.*

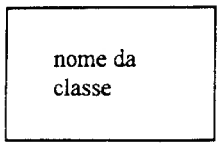
Cartão CRC (Classe, Responsabilidade e Colaboração):



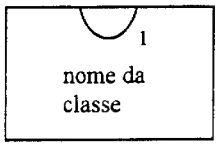
Cartão de subsistema:



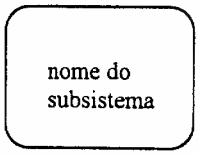
Classe:



Contrato:



Subsistema:



Colaboração entre classes:

Hierárquia de classes:

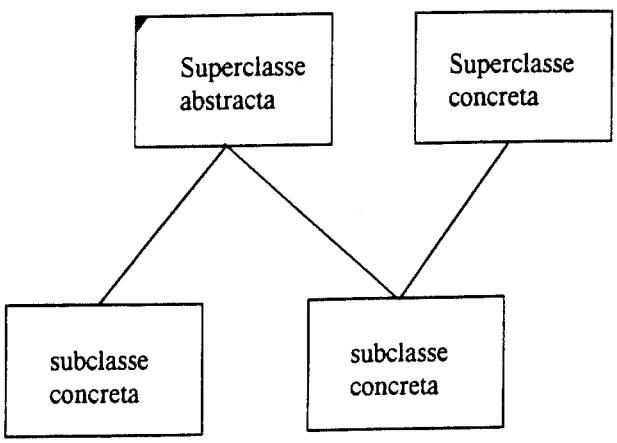


Diagrama de Venn:

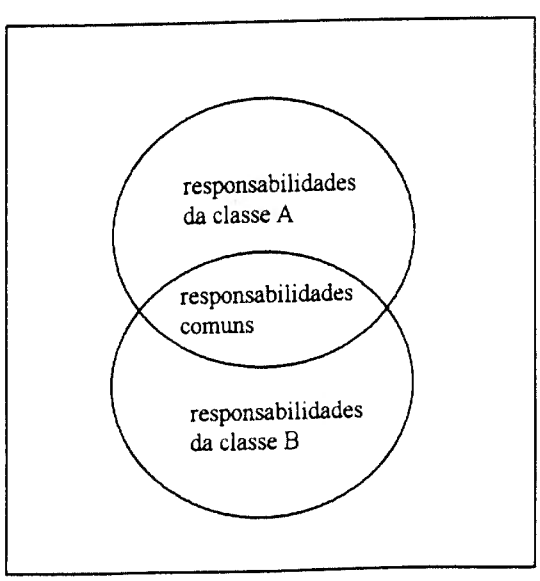
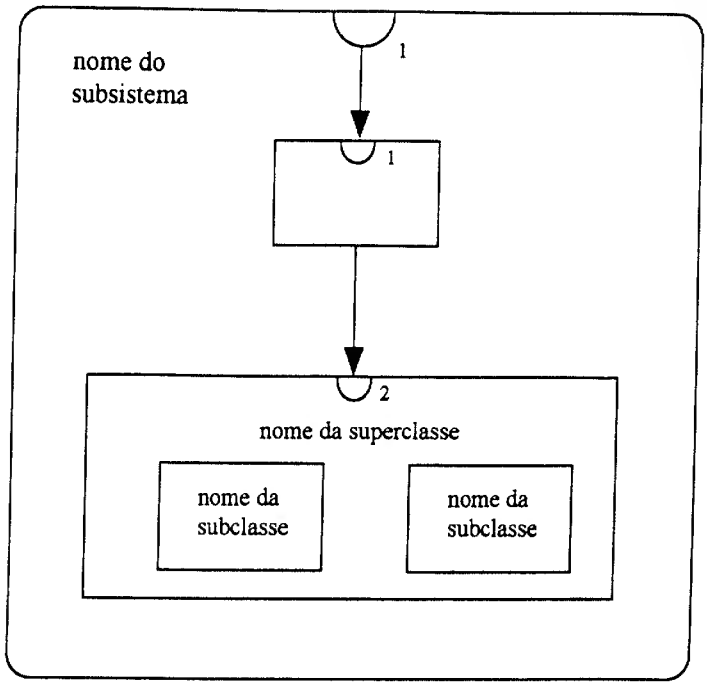


Gráfico de colaboração:



Especificação de uma classe:

Classe: nome da classe (concreta ou abstracta)

Superclasses: nomes das classes

Subclasses: nomes das classes

Gráfico hierárquico: página n°__

Gráfico de colaboração: página n°__

Descrição: descrição da classe

Contratos:

#. nome do contrato

responsabilidade

Responsabilidades privadas

responsabilidade

Página n° __

Especificação do subsistema:

Subsistema: nome do subsistema

Classes: lista de classes e subsistemas

Grafico de colaborações: página nº_

Descrição: descrição do subsistema

Contrato:

#. nome do contrato:

servidor: nome da classe ou do subsistema

Página nº __

Especificação do contrato:

Contrato # : nome do contrato

Servidor: nome da classe

Clientes: nome das classes ou subsistemas

Descrição: descrição do contrato

Página nº __

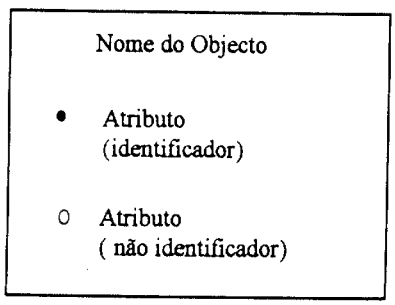
Análise Orientada para Objectos

Shlaer e Mellor

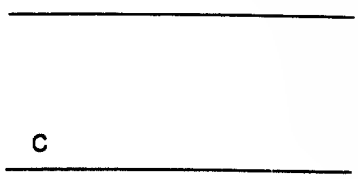
Modelo da Informação

(Diagrama de estrutura da informação)

Objecto:

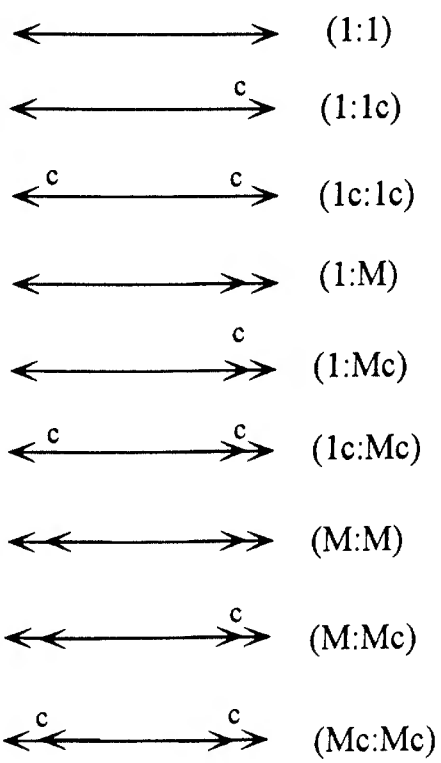


Relação:

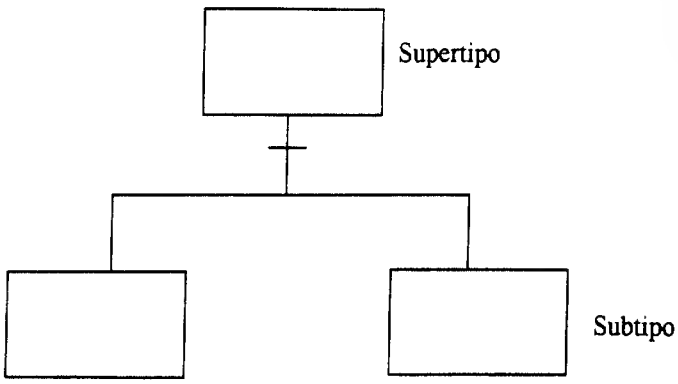


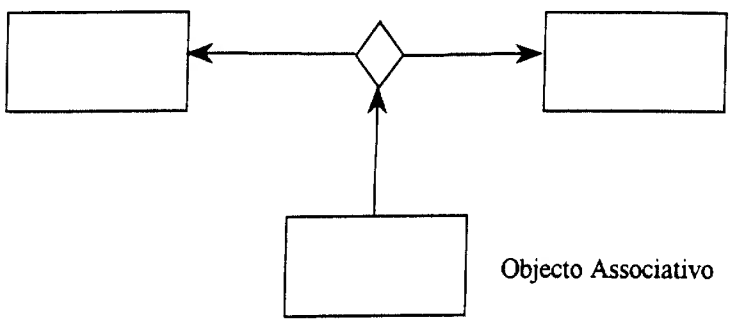
c - Relação condicional (podem existir instâncias que não participam na relação).

Grau da Relação



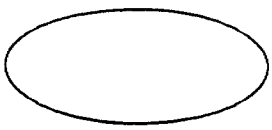
Generalização / Especialização



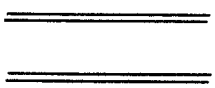


Modelo de Processos

(Diagrama de Fluxos de Dados)

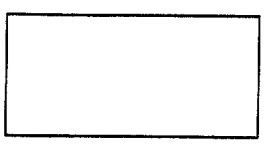


Processo



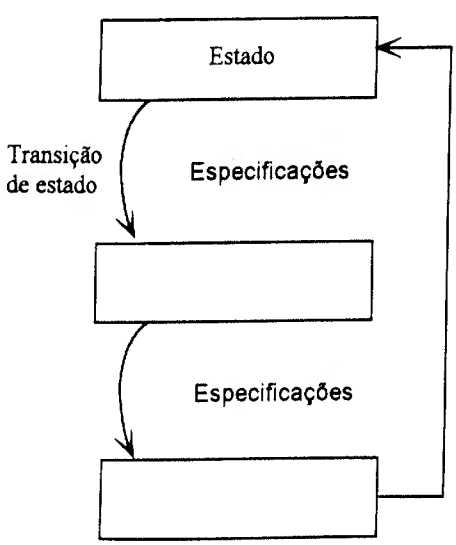
Arquivo

Fluxo de dados



Entidade Externa

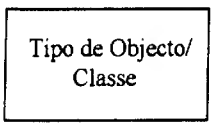
Modelo de Estados

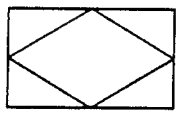


Análise Orientada para Objectos

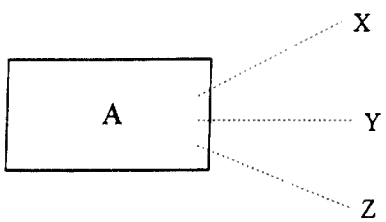
Martin e Odell

Esquema de Objectos





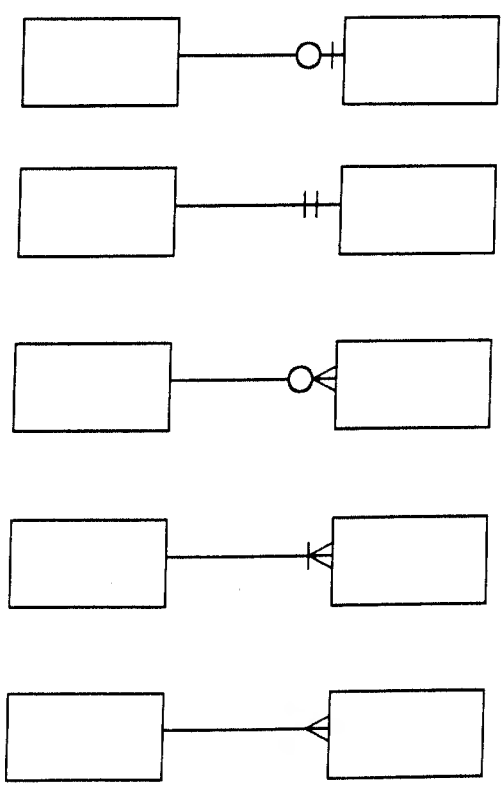
Intersecção
de dados



Instâncias

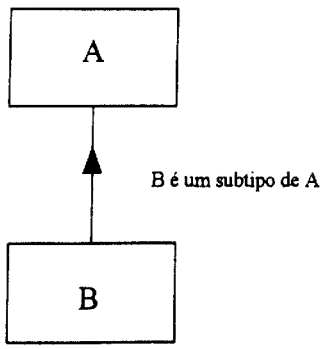
(X,Y e Z são objectos do tipo A)

Cardinalidade



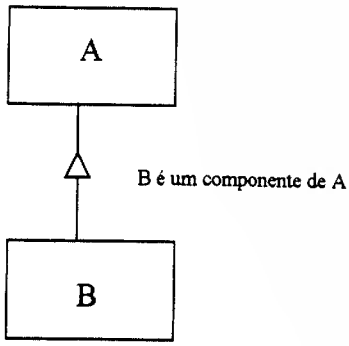
Máximo	Mínimo
0	1
1	1
0	+ de 0
1	+ de 1
+ de 1	+ de 1

Generalização

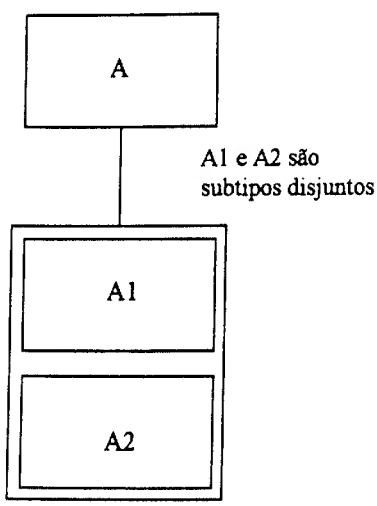


(subtipo individual)

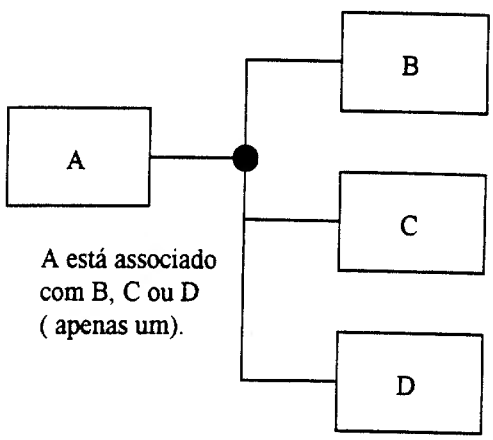
Composição



Supertipo / Subtipo



Mútua Exclusividade



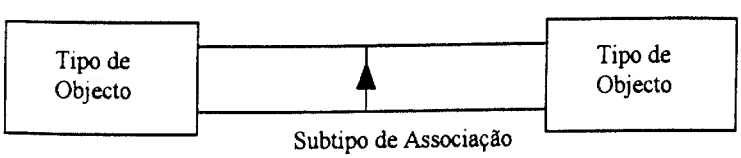
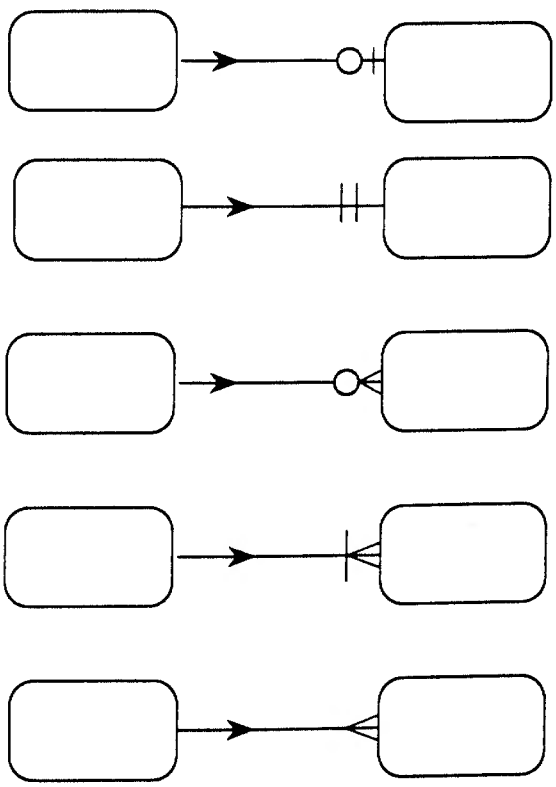


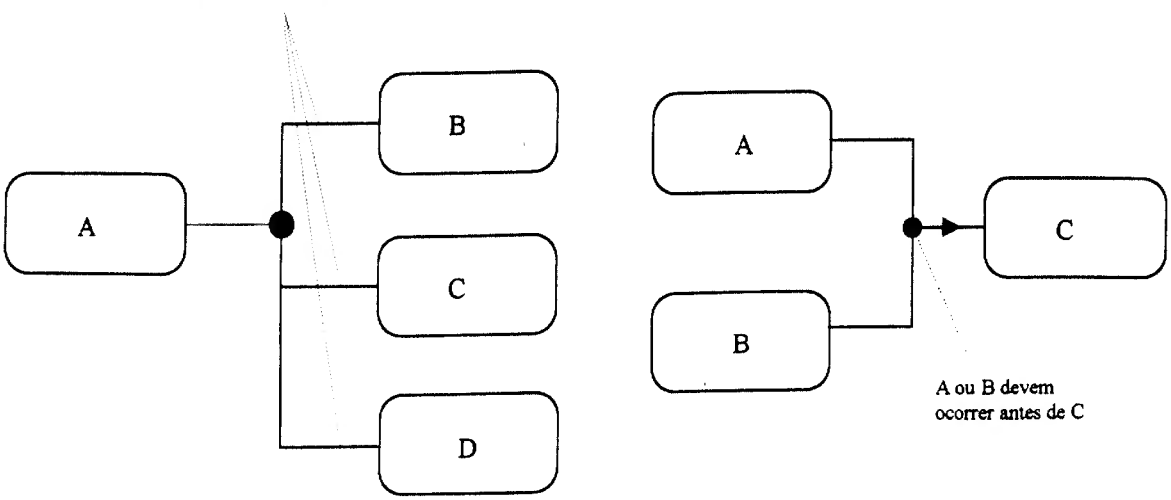
Diagrama de Dependência de Processos

Cardinalidade



Máximo	Mínimo
0	1
1	1
0	+ de 0
1	+ de 1
+ de 1	+ de 1

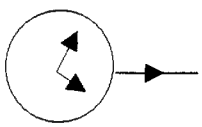
Condições mutualmente exclusivas



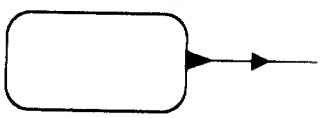
Esquema de Eventos

Tipo de Evento

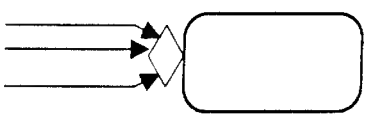
Tipo de evento temporal (*clock event type*)

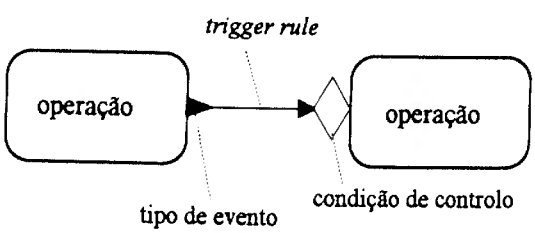


Operação resultado da ocorrência do evento

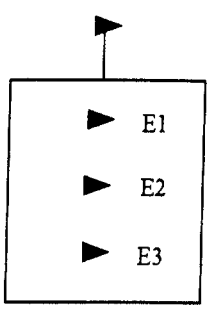


Condições de controlo
(na execução da actividade)





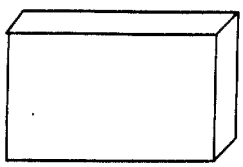
Tipo de evento E



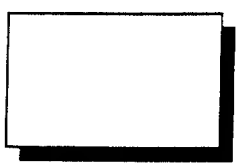
E1, E2 e E3 são subtipos disjuntos do tipo de evento E.

A ausência de uma linha vertical indica que esta partição não tem outros subtipos.

Diagrama de Fluxos de Objectos



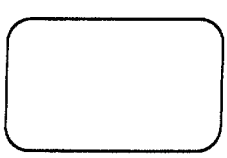
Objecto (representação fisica)



Tipo de objecto externo

Fluxo

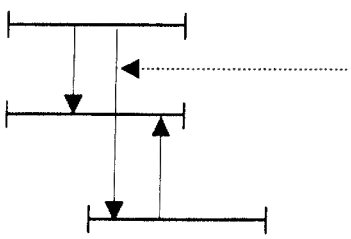
Representa precedência ou causualidade
(depende do contexo)



Actividade

Diagrama de Transição de Estados

Estado



Transição entre estados

SOM

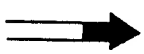
Velho

GSOM

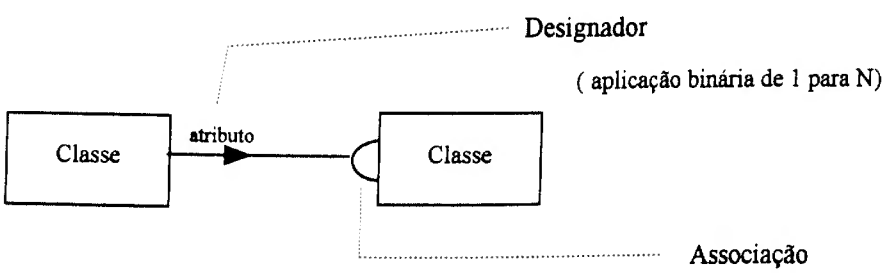
Classe

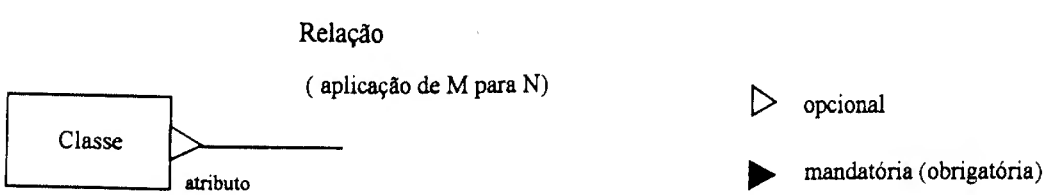
Fase

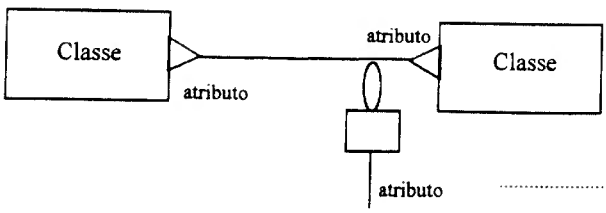
Classe básica (*boolean*, inteiro, carácter, real, monetária)



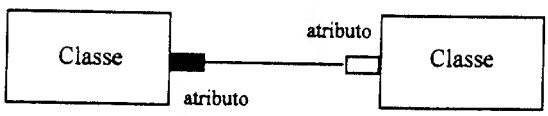
Ligação: fase origem - fase destino





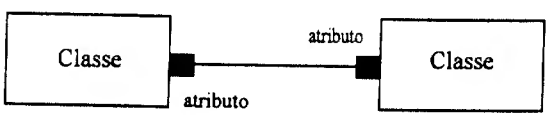


Atributo dependente

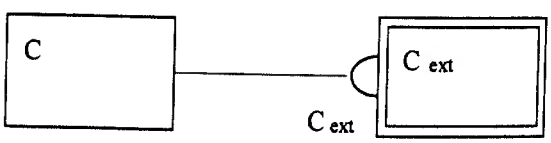


Agregação

agregado componente



Designador Duplo
(aplicação binária de 1 para 1)



Extensão de uma classe

OBLOG

Sernadas *et al.*

Diagrama do Universo das Classes

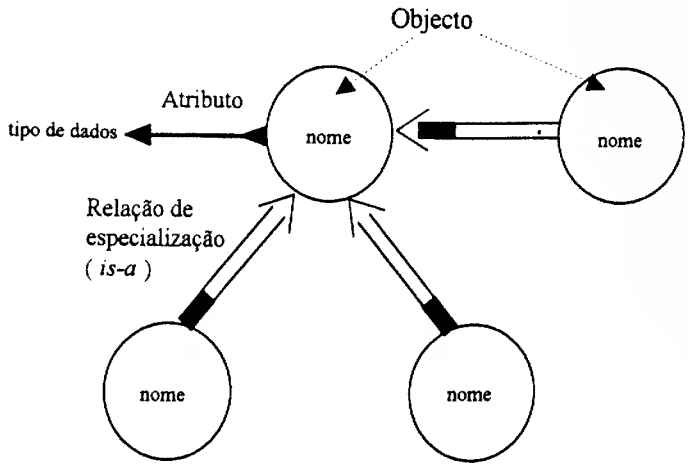


Diagrama Matricial de Classe

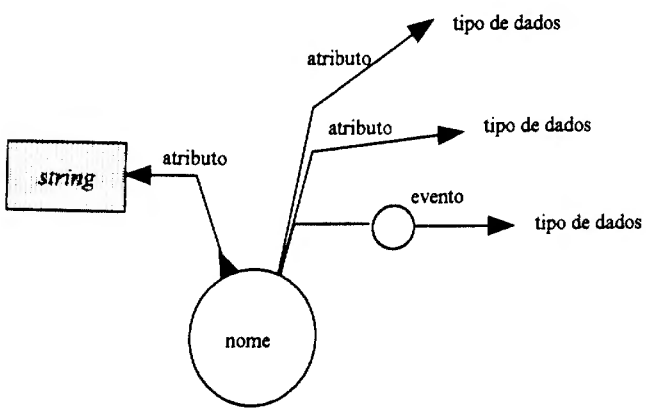


Diagrama de Comportamentos de Classe

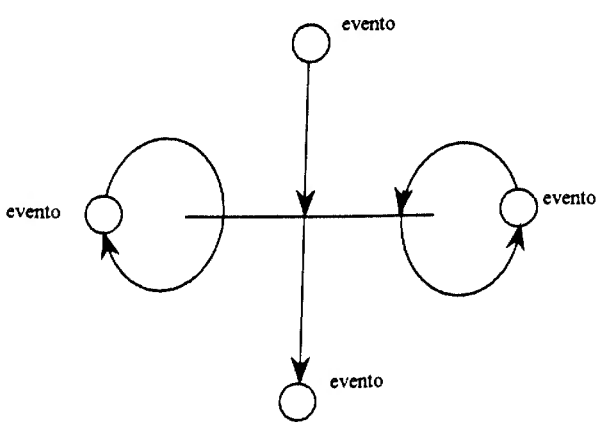


Diagrama de efeitos de evento sobre atributos

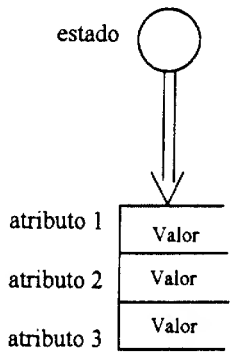
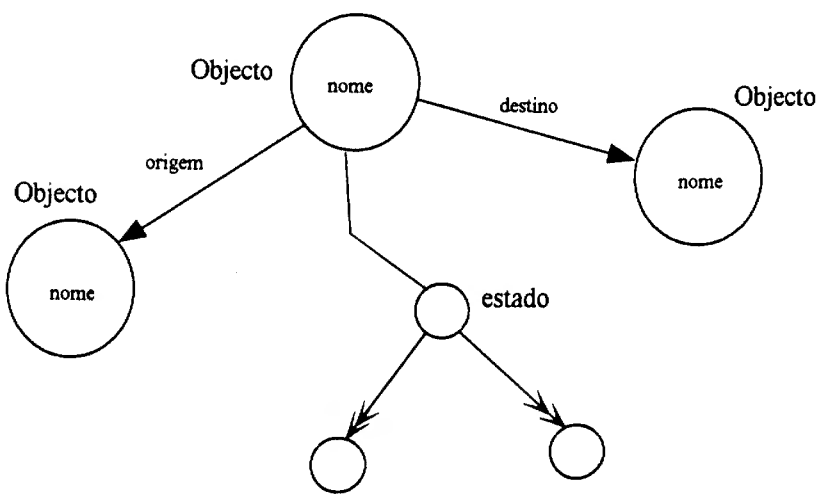


Diagrama de Interacção entre classes

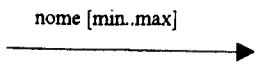


Objectory

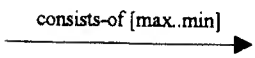
Jacobson *et al.*

Hereditariedade

Extensão



Associação

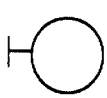


Associação de composição

Associação de comunicação

Actor

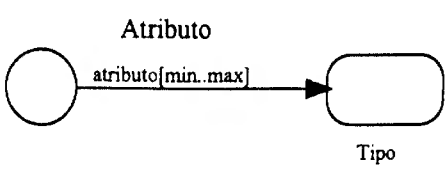
Acontecimento de utilização (*use case*)

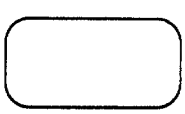


Objecto de *interface*

Objecto entidade

Objecto de Controlo





Subsistema